Identifying dog breeds based on visual variations in appearance, using Convolution Neural Networks & Transfer Learning

Dr. D Narayana
Professor, AIML,
Great Learning
Bengaluru, India
darapaneni@gmail.com

Debajyoti Das
Student, PGP-AIML, November 2019
Great Learning
Bengaluru, India
debajyoti.das.bookworm@gmail.com

Varun Sivasubramanian
Student, PGP-AIML, November 2019
Great Learning
Bengaluru, India
varunsiva95@gmail.com

Abstract **-- The main objective of this capstone project is to identify & classify the breed of a dog, based on the subtle changes in its visual appearance. The idea is to explore advanced computer vision techniques to intelligently & correctly predict the breed of a dog. Although most dogs confirm to certain visual traits that are unique to dogs, there are sometimes subtle & at time prominent features that help distinguish between different breeds of dogs. For example: While both Labradors & German Shepherds are dogs, there are clear visual demarcations that help a human intuitively discern between the two breeds. The example where may be overly simplistic, but in most cases, humans are able to tell between different breeds of dogs, if they have been pointed out to them. Humans are able to do so without even being told the distinguishable features between the breeds in most cases, for example between Labradors & German Shepherds. Human brain is able to intuitively understand this difference. In the current task we are trying to build a computer-vision based capability that helps computers to be also able to intuitively discern between dog breeds, without being clearly told what distinguishable features to look for. A further challenge is that while there are so many dog breeds, that even humans are at times confused as to which breed a particular dog belongs, both due to features between breeds being almost discernible as well as the number of breeds being extremely large. In this project, we also look to solve this problem using traditional computer vision approaches & transfer learning.**

In our approach, we have tried multiple convolution neural network (CNN) architectures, varying from a CNN built from hand to using popular architectures like VGG-16, ResNet-50 & MobileNet. We have also tried techniques like Image Augmentation to see if the model performs better, or at times to understand if we can mitigate over-fitting. Thus, the objectives of this project have been two folds – one, to be able to intuitively distinguish between different breeds of dogs using different CNN architectures; second, to be able to do so across a wide range of dog breeds (120 to be precise), even if some of the breeds have not been adequately represented as other breeds. The techniques have been tested using Stanford Dogs data set.

Keywords*: I*mage Classification, CNN, transfer learning, dog breed, Resnet 50, Mobilenet v2, VGG-16

I. INTRODUCTION

The initial idea behind this project originated from a Kaggle Playground Prediction Competition hosted in 2017[1]. In this playground competition, the candidates were provided with a strictly canine subset of ImageNet in order to practice fine-grained image categorization. The data-set had 120 breeds of dogs and a limited number training images per class. Each image had a file-name that is its unique id. The data-set comprised of 120 breeds of dogs. The goal of the competition was to create a classifier capable of determining a dog's breed from a photo.

Being pet-lovers, we took to this idea as to how we can use our knowledge on computer vision and build a model that would be able to overcome 2 challenges that currently plagues even humans when it comes to being able to differentiate between classes of the same object, based on sometimes subtle but often-times not so subtle visual variations, especially when the number of such distinct classes is quite large (possibly in hundreds) & when even remembering the different classes itself becomes a challenge.

A dog, is quite easy to identify & probably segregate from a cat or a cow or a buffalo & other domesticated animal. However, humans often times fail to correctly identify between a dog & other member of the canine family (like a fox) especially when the distinguishing features are quite similar. Being able to be discern between two dog classes can have immense implications in dog breeding, treatment, and other areas where different approaches need to be taken for different dog breeds. These techniques can then be probably extended to other animals also & help in geo-tagging & other activities [2].

In recent years, application of Machine Learning to provide objective and scalable solutions to solve such visual challenges have gained popularity especially with the success of various Deep Learning techniques applied to computer vision across domains [4].

Breed classification is an especially difficult task, even for human beings, for reasons mentioned above like subtle changes in visual features & possibly extremely large number of breeds [5]. This is not just true for dogs, but for most animals. This is also true for any such case where the number of distinct classes belonging to a single group is very large & there are often times not so discernible features to distinguish between classes. This is attenuated by the fact that available data-sets are susceptible to occlusions & various other optical challenges like lighting, angle, etc. & hence makes an already complicated task, even more challenging. Therefore, there are many challenges to be addressed and solved to integrate intuitive but accurate classification algorithms in such use cases.

The data-set used for this project was hosted by Kaggle, as part of their Kaggle Playground Prediction Competition in 2017. The data-set is a strictly canine subset of ImageNet & comprised of 120 breeds of dogs and a limited number training images per class. Each image was assigned a unique id and the same was tagged against the breed the dog in the image belonged to in a separate csv file.

File descriptions:

- train.zip - the training set, comprised of images of different dog breeds tagged with a unique id.
- labels.csv - mapped the breeds, for the images in the train set, with the corresponding image id.

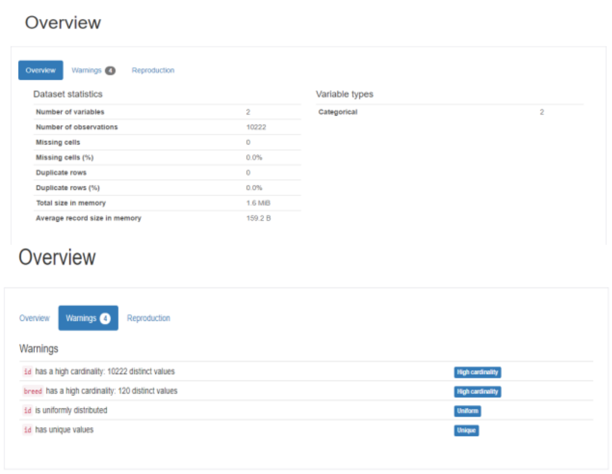
Kaggle authorities extend their gratitude to the creators of the Stanford Dogs Dataset [6], which made this competition possible: Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li.

II. MATERIALS & METHODS:

A. *Exploratory Data Analysis*

The data-set used for this project was hosted by Kaggle, as part of their Kaggle Playground Prediction Competition in 2017. The dataset was actually created as Stanford Dogs Dataset, and it is a strictly canine subset of ImageNet created for the task of fine-grained image categorization. It was created by Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. The dataset comprises of 120 breeds of dogs and a limited number training images per class. Each image was assigned a unique id and the same was tagged against the breed the dog in the image belonged to in a separate csv file. Below we mention the different file types:
File descriptions:

- train.zip - the training set, comprised of images of different dog, belonging to one of 120 breeds. Each image had a filename that is its unique id.
- test.zip – the testing set, where the model is expected to predict the probability of the image belonging to one of the 120 breeds.
- sample_submission.csv - a sample submission file in the correct format.
- labels.csv - mapped the breeds, for the images in the train set, with the corresponding image id.



Overview

Dataset statistics	
Number of variables	2
Number of observations	10222
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.6 MiB
Average record size in memory	159.2 B

Variable types	
Categorical	2

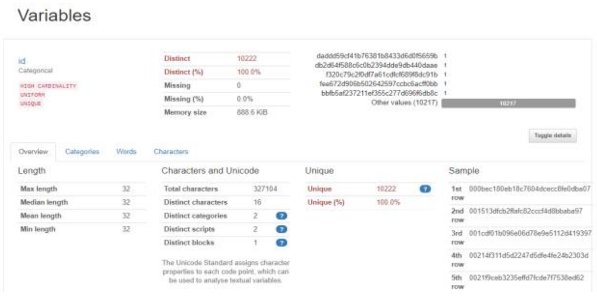
Overview

Warnings

id has a high cardinality: 10222 distinct values	High cardinality
breed has a high cardinality: 120 distinct values	High cardinality
id is uniformly distributed	Uniform
id has unique values	Unique

Fig. 1.: Overview of train.csv using pandas profiling

For our project, we have used only the train.zip & labels.csv files. as there was no way to evaluate the performance of the model on the test.zip, now that the Kaggle competition is closed. We have used pandas profiling for a quick EDA on the dataset.



Variables

id
Categorical
HIGH CARDINALITY
UNIFORM
UNIQUE

Distinct	10222
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	888.6 KiB

daddd59cf41b76381b8433d6d0f5659b	1
db2d64f588c6c0b2394dde9db440daae	1
f320c79c2f0df7a61cdfcf689f8dc91b	1
fee672d906b502642597ccbc6acff0bb	1
bbfb6af237211ef355c277d696f6db8c	1
Other values (10217)	10217

Length	
Max length	32
Median length	32
Mean length	32
Min length	32

Characters and Unicode	
Total characters	327104
Distinct characters	16
Distinct categories	2
Distinct scripts	2
Distinct blocks	1

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique	
Unique	10222
Unique (%)	100.0%

Sample	
1st row	000bec180eb18c7604dcecc8fe0dba07
2nd row	001513dfcb2ffafc82cccf4d8bbaba97
3rd row	001cdf01b096e06d78e9e5112d419397
4th row	00214f311d5d2247d5dfe4fe24b2303d
5th row	0021f9ceb3235effd7fcde7f7538ed62

Fig. 2: Variables section of pandas profiling used on train.csv

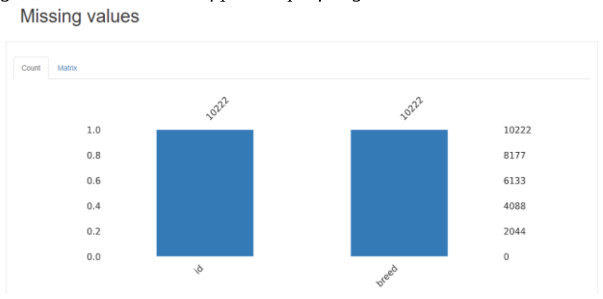


Fig. 3: Missing values section of pandas profiling used on train.csv

As we can see, there are 10222 total number of unique images. There are no NULL/Missing values or any duplicate values. The 'id' field comprises of unique id's assigned to each image and we have also showed some sample ids. The maximum length of the ids seems to be 32, which is consistent, i.e., all image ids are 32 characters long. There are also 120 distinct breeds, which we have mentioned earlier & is now confirmed in the above EDA too. Some sample dog -breeds are also displayed below

B. *Model*

We have predominantly used 80:20 split for training & validation sets, but in the last iteration we have also tried to build the model on 90:10 split. We have throughout used only 10000 of the total 10222 records for training & validation.

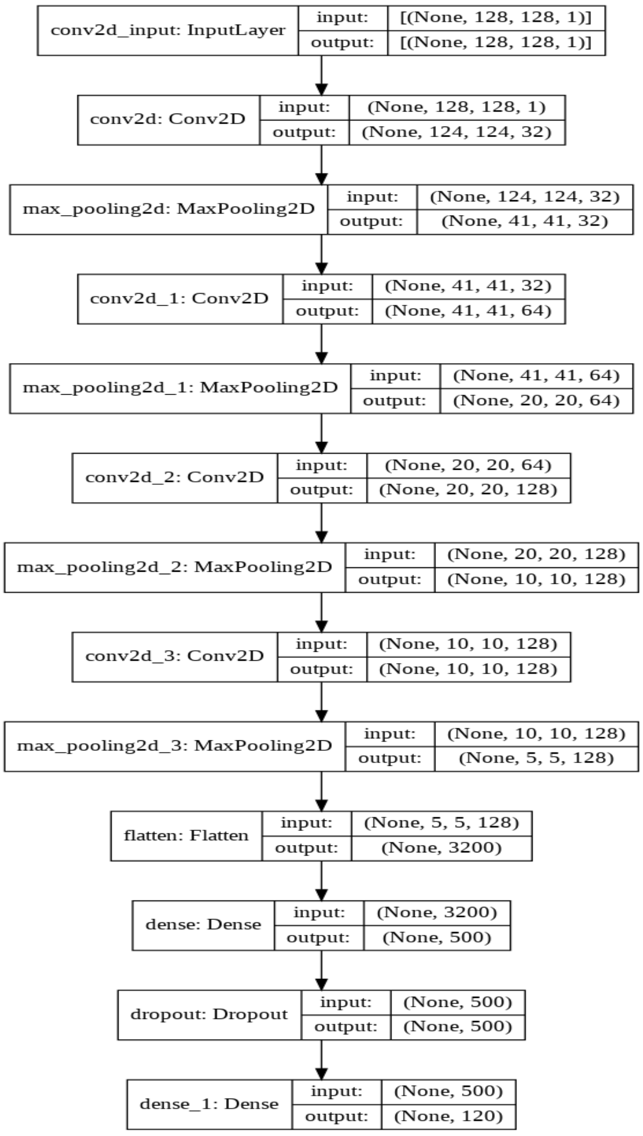
We have tried 3 types of image resizing techniques:

1. 128x128x1
2. 128x128x3
3. 224x224x3

We experimented with 4 types of models, namely, our custom CNN architecture, VGG-16, Resnet 50 and mobilenet v2.

1. *Custom CNN Architecture*: trained on 10222 images of size 128X128X1, split into an 80:20 split for training & validation, resulting in 8177 & 2045 images respectively. *Fig. 1:* shows the architecture of the custom CNN model.

We ran the model with or without Image Augmentation and recorded the accuracy in both cases.


conv2d_input: InputLayer | input: [(None, 128, 128, 1)] | output: [(None, 128, 128, 1)]
conv2d: Conv2D | input: (None, 128, 128, 1) | output: (None, 124, 124, 32)
max_pooling2d: MaxPooling2D | input: (None, 124, 124, 32) | output: (None, 41, 41, 32)
conv2d_1: Conv2D | input: (None, 41, 41, 32) | output: (None, 41, 41, 64)
max_pooling2d_1: MaxPooling2D | input: (None, 41, 41, 64) | output: (None, 20, 20, 64)
conv2d_2: Conv2D | input: (None, 20, 20, 64) | output: (None, 20, 20, 128)
max_pooling2d_2: MaxPooling2D | input: (None, 20, 20, 128) | output: (None, 10, 10, 128)
conv2d_3: Conv2D | input: (None, 10, 10, 128) | output: (None, 10, 10, 128)
max_pooling2d_3: MaxPooling2D | input: (None, 10, 10, 128) | output: (None, 5, 5, 128)
flatten: Flatten | input: (None, 5, 5, 128) | output: (None, 3200)
dense: Dense | input: (None, 3200) | output: (None, 500)
dropout: Dropout | input: (None, 500) | output: (None, 500)
dense_1: Dense | input: (None, 500) | output: (None, 120)
Fig. 4: Model plot of our custom CNN architecture

2. *Custom VGG16 model*: with pre-trained weights: VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [7].

The model achieves 92.7% top-5 test accuracy in ImageNet [8], which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes the improvement over AlexNet [9] by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. We trained the VGG16 model on 10222 images of size 128X128X3, split into an 80:20 split for training & validation, resulting in 8177 & 2045 images for each respectively.

We had also trained the model using Image Augmentation on top of the 10222 images of size 128X128X3, split into an 80:20 split for training & validation respectively with 32 images in each batch of augmented images.

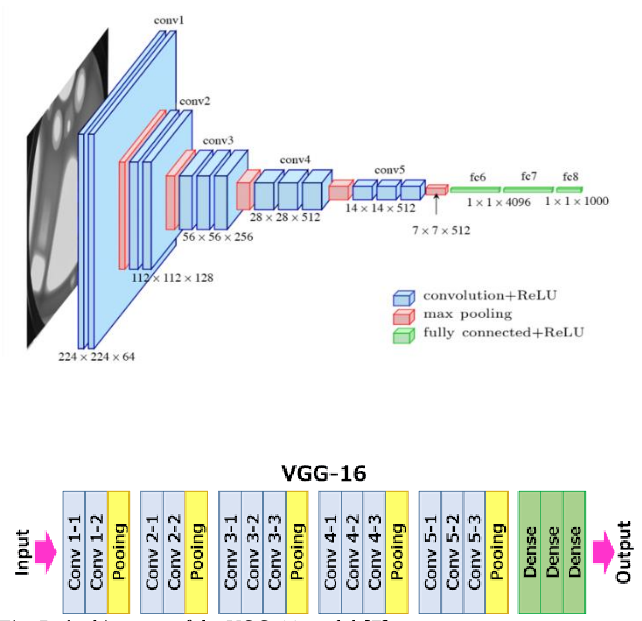


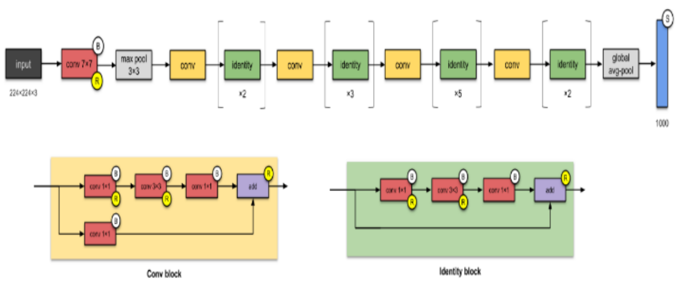
Fig. 5: Architecture of the VGG-16 model [7]

3. *Resnet50 pre-trained on ImageNet*: The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers [34]. The ResNet-50 has over 23 million trainable parameters [34].

We trained the Resnet50 model on 10000 images of size 224X224X3 each, split into an 80:20 split for training & validation, resulting in 8000 & 2000 images for each respectively.

On the second iteration, we have trained the model using Image Augmentation on top of the 10000 images of size 224X224X3, split into an 80:20 split for training & validation respectively with 32 images in each batch of augmented images. Finally, we kept the layers as trainable and trained on 10000 images of size 224X224X3, split into an 80:20 split for training & validation, resulting in 8000 & 2000 images for each respectively

Fig. 6: Representative architecture of the Resnet 50 model [34]



4. MobileNet V2 architecture pre-trained on ImageNet: MobileNet is a streamlined architecture that uses depth wise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for mobile and embedded vision applications [15].

The structure of MobileNet is based on depth wise separable filters. We trained the model on 10000 images of size 224X224X3, split into an 80:20 split for training & validation, resulting in 8000 & 2000 images for each respectively.

In our second iteration, we trained the model using Image Augmentation on top of the 10000 images of size 224X224X3, split into an 80:20 split for training & validation respectively with 32 images in each batch of augmented images. For the third iteration, we trained on 10000 images of size 224X224X3, split into a 90:10 split for training & validation, resulting in 9000 & 1000 images for each respectively.

Finally, we trained using Image Augmentation on top of the 10000 images of size 224X224X3, split into a 90:10 split for training & validation respectively with 32 images in each batch of augmented images.

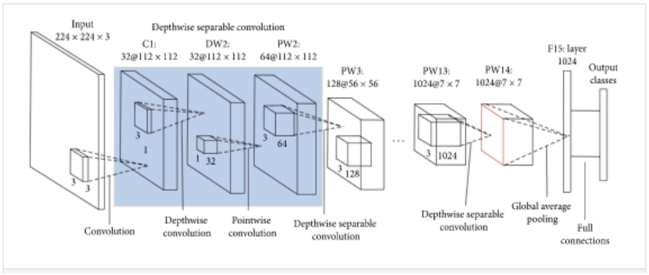


Fig. 7: Representative Architecture of the Mobilenet V2 model [15]

C. Optimization and Regularization

We have kept the parameters consistent across all of the above models, going for Adam as an Optimiser. We have also implemented call-backs to:

- Save model weights based on model accuracy, if there is no increase beyond 0.1%, over 5 iterations
- Reduce Learning Rate by a factor of 0.01 if there is no change (we have chosen the parameter as 'auto') in validation loss, i.e., no decrease in the same, by at least 0.01% over 5 iterations

To evaluate the models, we have looked at overall model accuracy & also at the traditional metrics like Precision, Recall & F1-scores across the classes. We have also focussed at top-5 accuracy, as this being a multi-class problem, we wanted to evaluate if the model was able to confirm the correct breed in 5 predictions, and the overall ROC-AUC score across the classes.

On the Regularisation front, we have looked at Image Augmentation to both add additional datapoints for the model to train & evaluate on, as well as a means to tackle over-fitting.

D. Loss Functions

We have used 'categorical cross-entropy' [20][21] as the loss function across all models, as the problem statement, i.e., minimization of multi-class log loss, has not changed across the models. The categorical crossentropy loss function calculates the loss of an example by computing the following sum:

$$Loss = -\sum_{i=1}^{output\ size} y_i \cdot log\widehat{y_i}$$

where $\hat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value, and output size is the number of scalar values in the model output.

This loss is a very good measure of how distinguishable two discrete probability distributions are from each other. In this context, y_i is the probability that event i occurs and the sum of all y_i is 1, meaning that exactly one event may occur. The minus sign ensures that the loss gets smaller when the distributions get closer to each other.

E. Training

As mentioned in the Data Partition section we created training and validation sets from the entire train.zip file & either split it in 80:20 or 90:10 ratio for training & validation respectively. We have demonstrated the model performance on 1 sample image pulled from the internet.

Pre-Processing: The images were resized based on the algorithm we were trying to evaluate. So, for the custom CNN architecture that we mentioned above, both with & without Image Augmentation, we resized the image to a grayscale format across 128X128 pixels. On similar lines, for the various VGG16 based models we used images of dimensions 128X128X3, i.e., colour images across 3 channels. Lastly, for the ResNet50 & MobileNet models we trained the models on images of dimensions 224X224X3.

In all the cases, we used cv2 for image extraction from the unzipped train.zip file. These extracted images were mapped against the dog breeds that were mentioned in the labels.csv file. The processed images were converted into arrays of datatype float32 and normalized by dividing by 255 (as image pixel can have a maximum value of 255).
For the 1st instance, we had to extend the dimensions as the models expected 3-d images, but our images by virtue of being grayscale were only 2-d.

Data Augmentation: When using Image Augmentation, we have referred to a standard set of augmentation techniques as below:

- width_shift_range=0.2,
- height_shift_range=0.2,
- rotation_range=30,
- shear_range=0.2,
- zoom_range=0.2,
- horizontal_flip=True,
- vertical_flip=True,
- fill_mode='nearest'

Model Training: The pre-processed images were fed to the models in batches. Either with, or without Image Augmentation, the model was always trained on batches of 32. In case of Image Augmentation, the augmented images were also generated in batches of 32 images each for training

& validation. All models were trained for 20 iterations. We also used call-backs like ModelCheckpoint & ReduceLROnPlateau.

These were parameterized such that Model weights would be saved, based on model accuracy, if there is no increase beyond 0.1%, over 5 iterations.The Learning Rate would be reduced by a factor of 0.01 if there is no change in validation loss (we have chosen the parameter as ‘auto’, so in the current context this will look to decrease the same) by at least 0.01% over 5 iterations

III. RESULTS

A. *Performance*

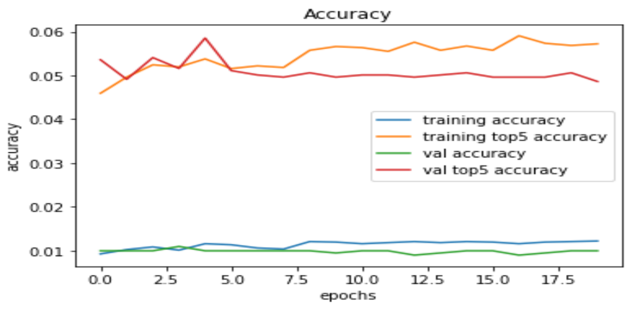


Fig. 8: Learning curves for the custom CNN model with Image Augmentation

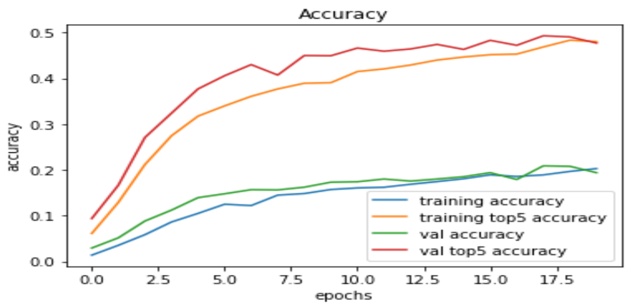


Fig. 9: Learning curves for the VGG-16 model with Image Augmentation

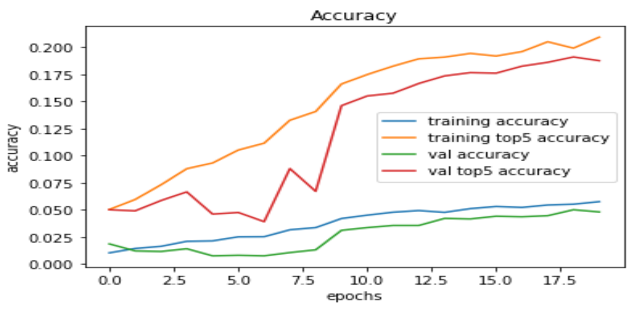


Fig. 10: Learning curves for the Resnet-50 model with Image Augmentation and trainable layers

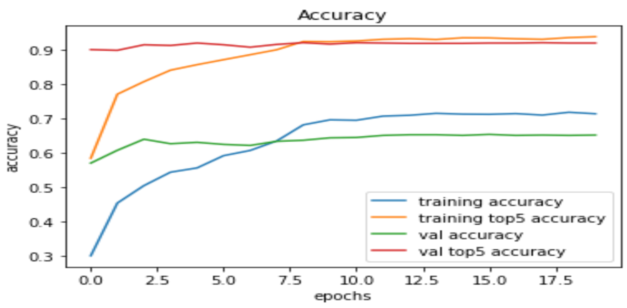


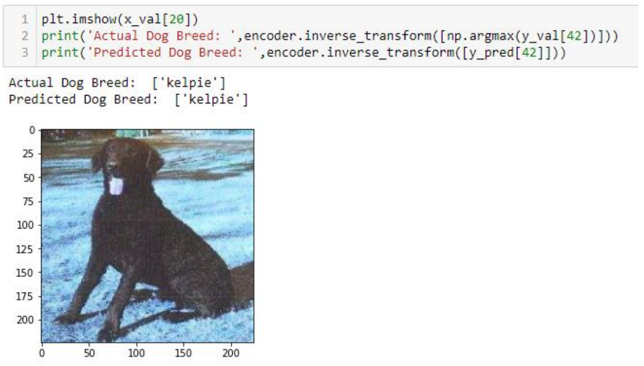
Fig. 11: Learning curves for the Mobilenet v2 model with Image Augmentation and trainable layers

Iterations	Model Name	Image Augmentation (Y/N)	Training Accuracy	Validation Accuracy	Val Top 5 Accuracy	ROC-AUC Score
1	Custom CNN Architecture	N	1.23%	1.08%	5.23%	0.50
2	Custom CNN Architecture	Y	1.24%	0.99%	4.86%	0.49
3	VGG-16 with pre-trained weights	N	99.96%	29.34%	60.54%	0.92
4	VGG-16 with pre-trained weights	N	54.76%	25.23%	53.89%	0.92
5	VGG-16 with pre-trained weights	Y	21.35%	19.39%	47.72%	0.89
6	Resnet-50 using pre-trained weights	N	1.15%	0.75%	5.05%	0.5
7	Resnet-50 using pre-trained weights	Y	1.09%	0.76%	5.04%	0.5
8	Resnet-50 with all layers trainable	N	5.92%	4.80%	18.75%	0.77
9	Mobilenet V2 with pre-trained weights	N	99.96%	71.40%	93.65%	0.99
10	Mobilenet V2 with pre-trained weights	Y	74.9%	65.78%	91.53%	0.99
11	Mobilenet V2 with pre-trained weights on 90:10 split	N	98.9%	67.10%	91.2%	0.99
12	Mobilenet V2 with pre-trained weights on 90:10 split	Y	71.62%	65.12%	91.94%	0.99

Table1: Accuracy and ROC score of models across 12 iterations

The preceding graphs and table summarize our observations from the 12 models. The custom CNN, VGG-16 and resnet-50 models showed a poor top-5 accuracy & overall roc-auc score, which are the 2 metrics that we would are looking at. The mobilenet v2 model which performs really well, doesn’t seem to be affected by increasing the Training data, but is much more generalized when we use Image Augmentation, and is not overfit. The model is also able to perform well, if we test it on random images extracted from the Internet.

All models have been run for 20 iterations, with a batch size of 32. Wherever Image Augmentation has been used, we have parameterized the generators to create batches of 32 images for both training & validation. This was done considering the memory capacity of Google Colaboratory which supports only 12.72GB of RAM & 2 CPUs from Intel (Xeon(R) CPU @ 2.00GHz 6th Gen) & 1 16GB Tesla P100-PCIE GPU on top of a 64GB Hard-Disk, when we choose a GPU based stack



```
plt.imshow(x_val[20])
print('Actual Dog Breed: ',encoder.inverse_transform([np.argmax(y_val[42])]))
print('Predicted Dog Breed: ',encoder.inverse_transform([y_pred[42]]))
```

```
Actual Dog Breed:  ['kelpie']
Predicted Dog Breed:  ['kelpie']
```

Fig. 12: The mobilenet v2 model correctly predicts the breed of the dog in the input image as Kelpie

B. Discussion

A lot of work has been done in the field of image recognition over the past few years. The goal is to build a model capable of identifying to which breed a dog belongs, based only on its photo. To train and evaluate the model, there are several factors that make the problem of dogs' categorization challenging.

Firstly, many different breeds of dogs exist, and all breeds on a high-level, look alike, i.e. there might only be subtle differences in appearance between some breeds. In fact, a dog's breed might be not obvious right away even for people who have an expertise in the domain. Therefore, determination of dog breeds provides an excellent domain for fine grained visual categorization experiments. Secondly, dog images are very rich in their variety, showing dogs of all shapes, sizes, and colors, under differing illumination, in innumerable poses, and in just about any location.

A simpler problem that is similar to dog categorization, is the problem of 'cats vs dogs' classification, and a lot of work has been done on it. A classifier [10] based on color features gets a 56.9% accuracy on the Asirra "Dogs vs Cats" dataset. Golle et al [11] achieved an accuracy of 82.7% using a Support Vector Machines classifier based on a combination of color and texture features. Researchers [12] used the SIFT (Scale-Invariant Feature Transform) features to train a classifier and finally got an accuracy of 92.9% on the Dogs vs Cats problem. With advancements of CNNs, the accuracy rates increased significantly. Thus, B. Liu et al achieved the accuracy of 94% by using CNNs to learn features of images, and SVMs for classification [13]. On the" Cats vs Dogs" Kaggle competition highest result of 98.9% was achieved by P. Sermanet in 2013 [14].

Liu et al [15] tried to tackle the dog breed identification problem using part localization. They built exemplar-based geometric and appearance models of dog breeds and their face parts. Their approach also featured a hierarchy of parts and breed-specific part localization. A recognition rate of 67% was achieved on a large real-world data set. Also, they experimentally demonstrated that accurate part localization significantly increases classification performance.

Tremendous progress has been made in fine-grained categorization with the advancement of deep CNNs. D. Steinkraus et. al [16] discusses data augmentation as an effective technique for improving the performance of CNNs when dealing with limited datasets. A paper [17] on transfer learning points out that deep learning models are by nature highly repurposable, and therefore a technique called transfer learning (transferring learned general knowledge from one, usually larger, problem to another one) is possible. It is emphasized that it is a very effective method for cases when only a small dataset is available for training.

In our iterations, we have tried building our own custom CNN architecture but were only able to achieve a top-5 accuracy of 5%. We then ventured into transfer learning as these models were extensively trained on millions of images and might perform better than our custom models. We observed that VGG-16 performed relatively well by achieving a top-5 accuracy of 60% in contrast to the Resnet-50 model that showed a top-5 accuracy of 19% (after making all its layers trainable). Both these models still performed much better than our custom CNN model with only a few layers. We can assume that a deeper architecture such as a VGG-16 is able to better extract higher level features that is required in such a fine-grain classification problem.

Finally, we saw that the more compact and lighter mobilenet V2 architecture, that was pre-trained, saw a remarkable top-5 accuracy of 94%. On the other hand, though, we did notice that the mobilenet V2 model was highly overfit, without image augmentation, with a staggering 30% difference between training accuracy & testing accuracy (comparing only top-1 accuracies here). However on using Image Augmentation on the training images, we see that there is very little overfit, as the difference between top-1 accuracies in the training and test data came down to about 7% (72% and 65% respectively). Thus, we can say that for this specific problem maybe the simpler mobilenet V2 strikes the right balance between complexity and performance and is able to efficiently extract the required high-level features to identify the breed of a dog from its image.

IV. CONCLUSION

Given the computational constraints, we had to be very careful about running the models. The models couldn't be executed serially as the image pre-processing required for each set of models, especially expanding the images as float32 arrays would occupy a lot of memory, and hence we had to execute the models in batches based on the image pre-processing required. We ran the custom CNN models together, and then ran the VGG16 based models together, finally, even the Resnet50 & MobileNet models had to be run independently as the former had a lot of memory requirement. Finally, since we decided to try the MobileNet model on a larger training-validation split (90:10), we had to run those 2 models separately.

We initially started with overall accuracy as an evaluation metric, but very soon shifted to roc-auc considering the diversity of the classes and skewness of distribution of records between them. We finally included top-5 accuracy as a benchmarking measure, as based on the above diversity of the classes and skewness of distribution of records, even a model which has a 90% plus top-5 accuracy could be potentially considered as an excellent solution candidate.

We have already mentioned earlier, that with a large number of classes and considering that the distinguishing features between some of the breeds maybe very fine visual differences, even humans find it difficult to be able to correctly identify & more importantly segregate between dog breeds. In such a circumstance, top-5 accuracy is a very good metric for model evaluation & bench-marking.

Most models, even if they were trained on ImageNet performed very poorly in all of the evaluation metrices. We even tried to retrain a pre-trained model, trained on ImageNet, of which the current dataset is only a sub-set, but the model performance didn't improve. Only the MobileNet V2 model pre-trained specifically for the current task, performed well.

V. REFERENCES

[1] "Dog Breed Identification," Kaggle.com. [Online]. Available: https://www.kaggle.com/c/dog-breed-identification/data. [Accessed: 13-Mar-2021].

[2] X. Wang, V. Ly, S. Sorensen, and C. Kambhamettu, "Dog breed classification via landmarks," in 2014 IEEE International Conference on Image Processing (ICIP), 2014.

[3] Vipul Jain and Dr. Bhoomi Gupta, "End-to-end multiclass dog breed classification," December 2020, vol. 6, no. 12, pp. 87–92, 2020.

[4] C. Wang, J. Wang, Q. Du, and X. Yang, "Dog breed classification based on deep learning," in 2020 13th International Symposium on Computational Intelligence and Design (ISCID), 2020.

[5] A. Konno, T. Romero, M. Inoue-Murayama, A. Saito, and T. Hasegawa, "Dog breed differences in visual communication with humans," PLoS One, vol. 11, no. 10, p. e0164760, 2016.

[6] "Stanford Dogs Dataset," Stanford.edu. [Online]. Available: http://vision.stanford.edu/aditya86/ImageNetDogs/. [Accessed: 13-Mar-2021].

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv [cs.CV], 2014.

[8] "ImageNet," Image-net.org. [Online]. Available: http://image-net.org/about. [Accessed: 13-Mar-2021].

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[10] J. H. J. Elson, J. Douceur, and J. Saul, "Asirra: a captcha that exploits interest-aligned 284 manual image categorization," 2007, pp. 366–374.

[11] P. Golle, "Machine learning attacks against the Asirra CAPTCHA," in Proceedings of the 5th Symposium on Usable Privacy and Security - SOUPS '09, 2009.

[12] V. A. Z. A. J. C. V. Parkhi and O. M, "Cats and dogs. in computer vision and 288 pattern recognition (cvpr," 2012, pp. 3498–3505.

[13] Y. L. K. Zhou B. Liu, "Image classification for dogs and cats," 2013.

[14] "Kaggle cat vs dog competition's public leaderboard," Kaggle. [Online]. Available: https://www.kaggle.com/c/dogs-vs290 cats/leaderboard. [Accessed: 13-Mar-2021].

[15] D. W. J. J. Liu, A. Kanazawa, and P. N. Belhumeur, "Dog breed classification using part 296 localization," 2012.

[16] D. S. P. Y. Simard and J. C. Platt, "Best practices for convolutional neural networks 298 applied to visual document analysis," 2003.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2010.

[18] P. Borwarnginn, W. Kusakunniran, S. Karnjanapreechakorn, and K. Thongkanchorn, "Knowing your dog breed: Identifying a dog breed with deep learning," Int. J. Autom. Comput., vol. 18, no. 1, pp. 45–54, 2021.

[19] R. O. Sinnott, F. Wu, and W. Chen, "A mobile application for dog breed detection and recognition based on deep learning," in 2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT), 2018.

[20] A. Rusiecki, "Trimmed categorical cross-entropy for deep learning with label noise," Electron. Lett., vol. 55, no. 6, pp. 319–320, 2019.

[21] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," arXiv [cs.LG], 2018.

[22] R. J. Simpson, K. J. Simpson, and L. VanKavage, "Rethinking dog breed identification in veterinary practice," J. Am. Vet. Med. Assoc., vol. 241, no. 9, pp. 1163–1166, 2012.

[23] A. Ayanzadeh and S. Vahidnia, "Modified deep neural networks for dog breeds identification," Preprints, 2018.

[24] M. V. Sai Rishita and T. Ahmed Harris, "Dog breed classifier using convolutional neural networks," in 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), 2018.

[25] D. D. Durga Bhavani, Professor, CVR College of Engineering/CSE Department, Hyderabad, India, M. H. S. Quadri, Y. Ram Reddy, PG Scholar, CVR College of Engineering/ CSE Department, Hyderabad, India, and Software Engineer, Eximius Design India Pvt. Ltd., Bengaluru, India, "Dog breed identification using convolutional neural networks on android," CVR Journal of Science & Technology, vol. 17, no. 1, pp. 62–66, 2019.

[26] S. Divya Meena and L. Agilandeeswari, "An efficient framework for animal breeds classification using semi-supervised learning and multi-part convolutional neural network (MP-CNN)," IEEE Access, vol. 7, pp. 151783–151802, 2019.

[27] X. Tu, K. Lai, and S. Yanushkevich, "Transfer learning on convolutional neural networks for dog identification," in 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018.

[28] C. L. Hoffman, N. Harrison, L. Wolff, and C. Westgarth, "Is that dog a pit bull? A cross-country comparison of perceptions of shelter workers regarding breed identification," J. Appl. Anim. Welf. Sci., vol. 17, no. 4, pp. 322–339, 2014.

[29] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in 2009 IEEE 12th International Conference on Computer Vision, 2009.

[30] B. Yao, A. Khosla, and L. Fei-Fei: "Combining randomization and discrimination for fine-grained image categorization," 2011.

[31] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar: "Attribute and simile classifiers for face verification," 2009.

[32] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in 2013 IEEE International Conference on Computer Vision Workshops, 2013.

[33] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," arXiv [cs.CV], 2013.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.