Dog Breed Identification

Akhil Nair, Satish Chirra

1 Introduction

Humans make vision seem easy. It doesn't take any effort for us to tell apart a cat and a dog, read a sign, or recognize a friends face. But these are actually hard problems to solve with a computer: they only seem easy because our brains are incredibly good at understanding images. Image classification is one of the classic problems in computer vision: given an image, identify it as being a member of one of several fixed classes. In this project we build a deep learing Convolutional Neural Network in R using Keras and Tensorflow. The model was then compared to a pre-trained image classification model trained on Imagenet[1].

1.1 Dataset

To build our image classification model, we used Stanford Dog dataset[2] which contains 20580 images of 120 different dog breeds where each image is annotated with a object class label. The dataset is extremely challenging due to a variety of reasons. First, being a fine-grained image categorization problem, there is little inter-class variation.For example the Siberian Husky and Alaskan Malamute share very similar characteristics. Second, there is a very large intra-class variation i.e. images within a class have different ages, colors and poses. Third, variations in background of the images.

2 Pre-Processing

The first step as part of pre-processing was to re-size the images to 64*64 pixels. This is to have uniformity in the size of images and also considering the computational restrictions. The next step was to add a little bit of variance in our data since the images in the dataset were very organized and contained no noise. As we want our model to classify images with noise we artificially added noise by a random combination of zoom and horizontal flip. The dataset was then split into train and test. As this is a classification task we took into consideration the proportions of our target classes and made sure the train set has equal number of images (100) from each class resulting in 12000 images in train set and the remaining 8580 images in the test set.

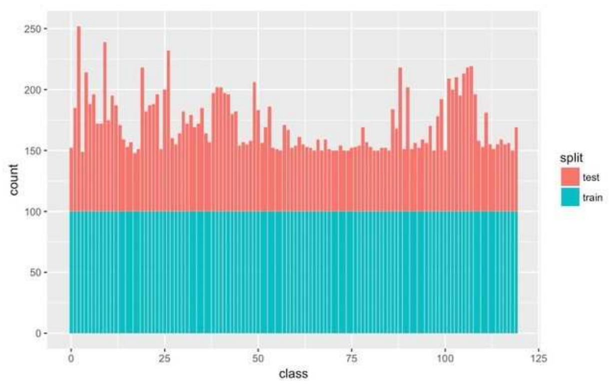

Fig. 1. Train/Test Split

3 Exploratory Data Analysis

To have an understanding on the distribution of the breeds in the dataset, we found out the most popular and rare breeds. The most popular breed is Maltese. Below are the top five popular breeds.

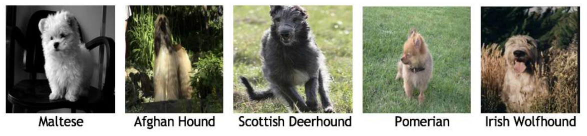

Fig. 2. Popular Breeds

The rarest breed is Redbone. Below are the top five rare breeds.

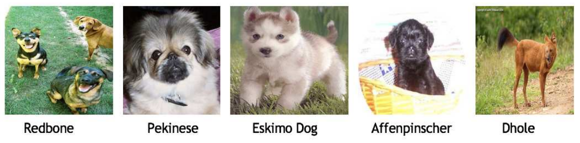

Fig. 3. Rare Breeds

4 Methods

4.1 Convolutional Neural Network (CNN)

CNN is a category of deep artificial neural network known to be effective on image classification. The CNNs have several different filters consisting of (randomly initialized) trainable parameters depending on the depth and filters at each layer of a network, which can convolve on a given input volume (the first input being the image itself) spatially to create some feature/activation maps at each layer. During this process, (through backpropagation) they learn by adjusting those initial values to capture the correct magnitude of a spatial feature on which they are convolving. These high number of filters essentially learn to capture spatial features from the input volumes based on the learned magnitude. Hence they can successfully boil down a given image into a highly abstracted representation which is easy for predicting[3].

4.2 CNN Architecture

In this section we describe the architecture of our model. A CNN has three layers. The first is a Convolutional layer where a sliding window (filter) is used to extract features from the image. Considering the non-linear nature of real world images we introduced non-linearity into our model by passing a Relu function into the convolutional layer. The second layer is a Pooling layer, that reduces the dimensions, which makes the model efficient and prevents over-fitting. We used max pooling as it has shown to work better in practice. The final layer is a Fully Connected layer that uses a softmax activation function in the output layer. The purpose of the Fully Connected layer is to use the features extracted by the previous layers to classify the input image into various classes based on the training dataset.

Layer	Filter Size	Volume Size
Input	N/A	3*64*64
Convolution	(3,3)	(12000, 32, 62, 62)
Convolution	(3,3)	(12000, 32, 60, 60)
Max Pooling	(2,2)	(12000, 32, 30, 30)
Convolution	(3,3)	(12000, 64, 28, 28)
Convolution	(3,3)	(12000, 64, 26, 26)
Max Pooling	(2,2)	(12000, 64, 13, 13)
Convolution	(3,3)	(12000, 128, 11, 11)
Convolution	(3,3)	(12000, 128, 9, 9)
Max Pooling	(2,2)	(12000, 128, 5, 5)
Fully Connected	N/A	(12000, 512)
Drop Out	N/A	(12000, 512)
Fully Connected	N/A	(12000, 120)
Softmax	N/A	(12000, 120)

In our model architecture we have three sets of convolution, relu and pooling layer where each set comprises two convolutions. The first set uses 32 filters of 3*3 size and a 2*2 max pooling. The second set introduces 64 filters of 3*3 size and a 2*2 max pooling. The last set has 128 filters of 3*3 size and a 2*2 max pooling. Together these layers extract useful features from the images, introduce non-linearity in our network and reduce feature dimension. We then have two fully connected layers in our architecture to classify our images based on the features extracted. We included a dropout layer to the first fully connected layer which has 512 neurons to avoid over-fitting. The second fully connected layer has 120 neurons to output 120 probabilities one for each of our classes (dog breeds).

4.3 InceptionV3

In contrast to the above, we have used a pre-trained model (InceptionV3) to have a better understanding on how a model performs given the required computational power. The Inception micro-architecture was first introduced by Szegedy et al. in their 2014 paper[4]. The key insight is to realize that conventional convolutional ”filters” can only learn linear functions of their inputs where as Inception connects convolutional layers through multi-layer perceptrons that can learn non-linear functions. These perceptrons are mathematically equivalent to 1x1 convolutions, and thus fit neatly within the CNN framework. InceptionV3 is trained on 'Imagenet' dataset[1] which has 1.4 million images of 1000 different classes.

We have taken the pre-trained InceptionV3 model excluding the top layer, and fine-tuned it on our set of classes (120) by adding a 'Relu','Softmax' layer on top of it. In order to input the images into a InceptionV3 model we have converted the test images to 128*128 pixels. Once the images are converted and model weights are freezed, we used the model to predict the accuracy on test dataset.

5 Results

5.1 CNN

Our implementation gave us an accuracy of 17 percent on the test set for 1000 iterations. The system in the original paper gave an accuracy of 22 percent.

The low accuracy on the test set can be attributed to two main reasons. First, having 20000 images of 120 different breeds is not enough to train a deep neural network. Even though convolutional neural networks is by all account the best model for image classification, but in our case there isn't enough images in the train set to train. This

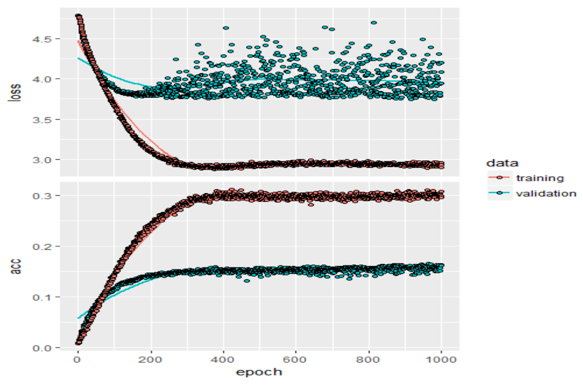

Fig. 4. Accuracy using our CNN Model

causes the model to not learn generic enough patterns of the dataset to classify different dog breeds. Second, we have sacrificed the image size by resizing our images to 64*64 pixels considering the computational restrictions.

5.2 Inception

To validate our above reasoning for low accuracy we ran our test set images through a pre-trained model (InceptionV3) which was trained on a much larger dataset (Imagenet) using images of size 299*299 pixels. This gave us an accuracy of 99.16 percent.

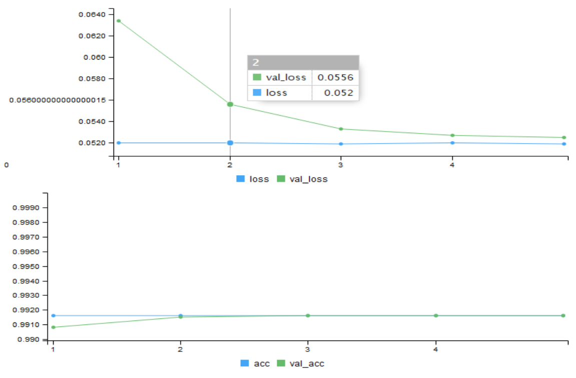

Fig. 5. Accuracy using Inception Model

6 Discussion

There are many things that we have learned from doing this project. First, we have designed a Convolutional Neural Network from the scratch and trained it using R. This is the major take away, because we always wanted to develop a deep neural network using R. The challenging part was to train the data which almost took two days for 1000 iterations. Secondly, we have learned that accuracy of a model depends on a lot of factors starting from the Architecture design of the model, computational power, volume of our data, learning rate, noise and in our case the resolution of the images. Tuning all the factors to have a best model is challenging. Thirdly, using pre-trained models to have better understanding on how the model performs when it is trained on a larger dataset like Imagenet.

In future we would like to run our model with a higher image resolution or of original size given the computational power. This would help us understand how the accuracy varies from the existing one. Also we would like to see how Reinforcement learning might help in getting a better accuracy. We would also like to use our model to create a dog breed prediction app using R-Shiny.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. Inv CVPR, 2009.

[2] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao and Li Fei-Fei.Novel dataset for Fine-Grained Image Categorization.First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[3] AnirbanRay. Do-convolutional-neural-networks-work-better-on-image-classification-problems-than-recurrent-neural-networks

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich.Going Deeper with Convolutions,2015.

[5] Abdellatif Abdelfattah.Image Classification using Deep Neural Networks-A begineer friendly approach using Tensorflow

[6] CvTricks Tensorflow. Training Convolutional Neural Network for Image Classification

[7] Ujjwal Karn. An intuitive Explanation of Convolutional Neural Networks