Dog Breed Classifier via Convolutional Neural Network

Cheng-Chieh Peng

1. Introduction

Can a machine think and act like a human being, which can pass the Turing test so people are not able to tell is it a machine? It should possess all the abilities people can do like listening, watching, and have a meaningful response after it digests the information he receives. That's a goal for artificial intelligence which people are eager to achieve.

Computer vision is one of the important subfields. Computers record what they receive by pixels with colors as current technology. How to let the computer know what does it see, it needs to find the border, grouping pixels to a different object, and then recognize. This was thought to be extremely hard, but thanks to the recent development deep learning technique, we are making big progress on the road.

The early training recognition tasks like handwriting numbers (MINST [1]) and the 10 classes of fashion style images (fashion-MINST[1]) in grayscale without background noise are achievable to be above 95% accuracy with shallow machine learning. People are now trying to solve more practical problems, which consist of colors and background, and also suffer the complexity from light the different angles of pictures as the problem for this project: the classification of dog breed. Different breeds of dogs might resemble in many ways, and even the same breed of dog could have different color make this problem challenge. We will build a solution based on machine learning modals to let computer output the breed from what it read.

3. Datasets and Inputs

The dataset is provided by Udacity. It consists of 133 dog breeds (each with more than 20 images) and a total of 8351 images, 80% of them are used for training, 10% for validation and 10% for test the performance. To utilize pre-trained deep learning Convolutional Neural Network

(CNN) models, each image is standardized for transfer learning with size 224*224 pixel, RGB color normalized to mean=[0.485, 0.456, 0.406] and standard deviation=[0.229, 0.224, 0.225].

4. Learning methods

The input image would be recognized as it either contains dogs or not via OpenCV's implementation of Haar feature-based cascade classifiers [2]. For dog images, it is identified via either a CNN model from scratch or some other pre-trained models.

The CNN model a kind of special deep learning neural networks which is composed of convolutional layers, pooling layers, and regular fully connected layers. The convolutional layer is a sophisticated way to reduce the number of parameter in deep learning (and therefore reduce the cost of computation) by applying matrix filter on layers to extract different features.

In this project, 3 pre-trained models will be used : VGG [3], DenseNet [4] and ResNeXt [5]. The pre-trained models are available from Pytorch official site. For deep learning neural network training, the number of parameters could be a huge number (easily above millions depends on design of model), thus the computational cost for training is expensive. Using the pre-trained models, which not only the neural network structures have been determined but also the parameters have been trained to optimal value, as the foundation of transfer learning has been proven to be successfully solving classification problem while saving training cost.

All the training is done through Google Colaboratory, which provide the free computational resources with GPU available to speed up the training processes.

4.1 CNN model from scratch

The first approach is built up a CNN model from scratch. This model is designed to be simple with only a few layers of networks to limit the usage of computational resources for training. The model includes 3 convolutional layers and two fully connected layers. The convolutional layers are applied to filter the features of the images. The pooling layers are applied after each convolutional layers to reduce the dimension and parameters. The fully connected layers stands for a layer which connected each inputs via different weighting and is

used for classification. Dropout layers, which randomly drop a pre-defined ration of network connections is applied to prevent overfitting on training sample.

4.2 Transfer CNN model

Three different pre-trained CNN models are used in this project. Due to the huge computational cost, the training is done with only the parameters at the last layer for classification are left float while the others are keep fixed with its pre-trained value.

VGG: The VGG is developed by Visual Geometric Group from Oxford University. There are some different variations of this model, which differ by the number of their weighting layers. In this project, the 16 weighting layer version (VGG-16) is used. The 16 weighting layers are composed of two parts: 6 groups of convolutional layers each consist of 2 or 3 convolutional layers and a max-pooling layer, and 3 fully connected layers. For its convolutional layers, it using 3 by 3 filters.

DenseNet: The DenseNet is stands for “Densely Connected Convolutional Networks”. Its basic idea is that instead of passing the information only to adjacent layer, it connected each layers. All subsequent layers will using all previous layers result as input. This idea is proposed to improve the architecture of deep learning to require less computational cost. In this project, the 121 version Densenet-121 is used.

ResNext: ResNext inherited the architecture of ResNet [6], in which can have a very deep network by introducing shortcut connection as residual learning. On top of it, ResNext proposed a strategy called “cardinality”, which perform split-transform-merge operation similar to what Inception model [7] but in a simple way. This is done by split the network into 32 same topology blocks. In this project, the 50 layers version ResNext50_32x4d is used.

5. Results, Benchmark Model

The performance of classification problem is usually measured by accuracy: number of correct predicted cases / number of all cases, or equivalently top1 error : 1 – accuracy. Another frequently used metric is top5 error, means that the percentage of cases its class is not in any of the top 5 predicted result. To limit the usage of computation and make a fair comparison, each

training method are performed with 20 epochs. The training result are shown in Table 1 with a comparison to result of pre-trained model on ImageNet 1000 class. Although in this project, the models have only trained to additional 20 epoch for classification, the performance is comparable to the ImageNet-1k result. Surprisingly, among those three pre-trained model, VGG-16 which has the simplest structure of networks and worst performance in ImageNet classification, achieve the best in dog breed classification. Its top-1 error is 13.4% while the top5 error is only 1.1%.

Table 1: Performance of different CNN models for dog breed classification and ImageNet-1k.

	Top-1 error (%)	**Top-5 error (%)**	**ImageNet-1k top-1 error (%)**	**ImageNet-1k top-5 error (%)**
CNN Model from Scratch	84.6	56.7	N/A	N/A
VGG-16	13.4	1.1	28.1	9.4
DenseNet-121	29.1	5.6	25.4	7,8
ResNext_32x4d	23.3	3.5	20.4	5.3

For VGG-16 model, there are 3 breeds of dog cannot be well identified: German wirehaired pointer, Norwegian elkhound and Pointer. The top-5 error of them are above 25% and top-1error are above 50%. These 3 breeds cannot be well identified by other models either. It might implies that the misidentification is either due to the limitation of CNN method or from the input images. Taking a close look at the worst case: German wirehaired pointer. In the top-3 predictions, two images are predicted as either English springer spaniel or Brittany shown in Figure 1 and two images are predicted as either Otterhound or Wirehaired pointing griffon shown in Figure 2. By checking the image manually, the dogs with its misidentified breeds are actually very similar, the difference between two sets of same breed dog (Figure 1 vs. Figure 2), German wirehaired pointer, seems even larger than the difference between it to its misidentified breed.

Figure 1: Test dog images: German wirehaired pointer (top 2 images) and their predicted breeds example images, Brittany (bottom left) and English springer spaniel (bottom right)

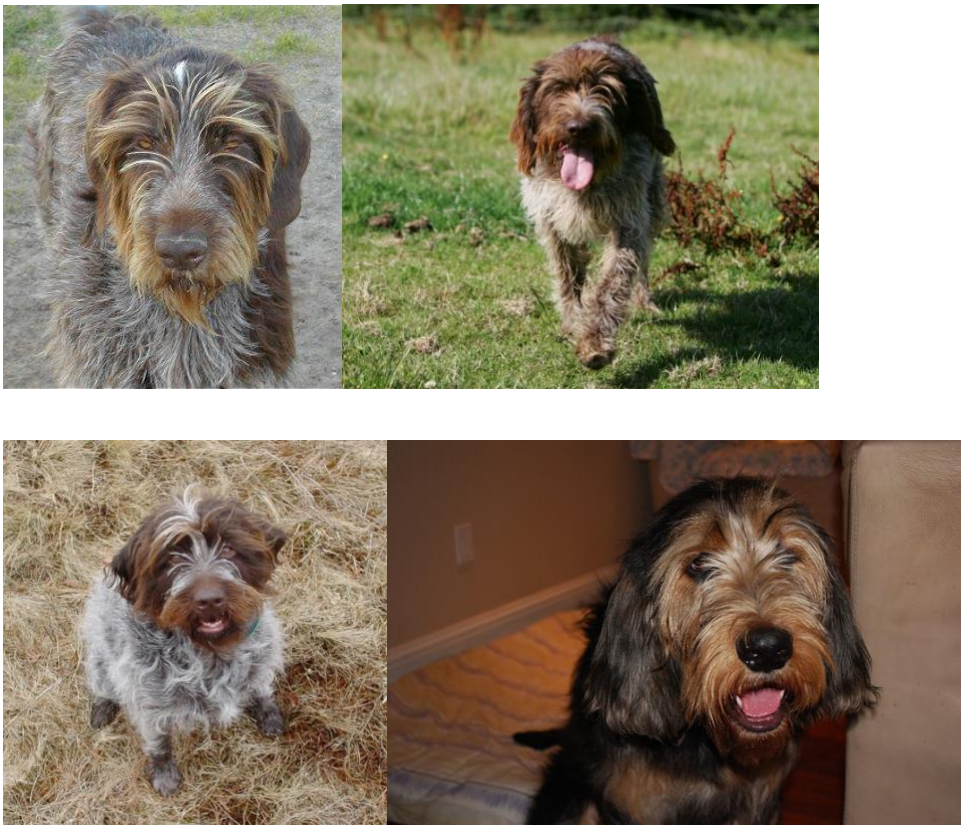

Figure 2: Test dog images: German wirehaired pointer (top 2 images) and their predicted breeds example images, Otterhound (bottom left) and Wirehaired pointing griffon (bottom right)

8. Conclusion

In Summary, we have built a CNN model from scratch and used three different pre-trained models: VGG, DenseNet and ResNeXt, for dog breed classification. With a reasonable amount of computing resources used for training (20 epoch for each), the model from scratch shows a 15.4% accuracy, is much higher than random guess (<1%). Pre-trained models achieved similar performance to their accuracy in ImageNet 1K classification (70-80%). Among those three models, VGG-16 achieved the best accuracy for top-1 error 13.4% and top-5 error only 1.1%. Checking with those dog images could not be correctly identified, the reason is due to the highly similarity of different breeds and limitation of a single image.

[1] ‘THE MNIST DATABASE of handwritten digits’, http://yann.lecun.com/exdb/mnist/
[2] ‘Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection.’ In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–900. IEEE, 2002.
[3] ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’, Karen Simonyan and Andrew Zisserman, arXiv:1409.1556
[4] ‘Densely Connected Convolutional Networks’, Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, arXiv: 1608.06993
[5] ‘Aggregated Residual Transformations for Deep Neural Networks’, Saining Xie and Ross Girshick and Piotr Dollár and Zhuowen Tu and Kaiming He, arXiv:1611.05431
[6] ‘Deep Residual Learning for Image Recognition’, Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, arXiv: 1512.03385
[7] ‘Rethinking the Inception Architecture for Computer Vision’, Christian Szegedy and Vincent Vanhoucke and Sergey Ioffe and Jonathon Shlens and Zbigniew Wojna, arXiv: 1512.00567
[8] ‘TORCHVISION.MODELS’, https://pytorch.org/docs/stable/torchvision/models.html