Dog Breeds Classification

CS5330 Final Project

Yueyang Wu
Khoury College of Computer Science
Northeastern University
San Jose, US
wu.yuey@northeastern.edu

Yuyang Tian
Khoury College of Computer Science
Northeastern University
Seattle, US
tian.yuya@northeastern.edu

Liqi Qi
Khoury College of Computer Science
Northeastern University
Seattle, US
qi.l@northeastern.edu

***Abstract*—Recognizing various dog breeds is essential in understanding the dog's characteristics and health conditions. Convolutional Neural Network(CNN) is a commonly used approach for image classification problems. In this project, we loaded and modified three pre-trained models to build and explore different dog breed classification models. The results show that the VGG16 model performs the best with a 75.9% accuracy. Also, MobileNet, as a lightweight model, trains and computes the fastest among the three models while maintaining a satisfying performance.**

I. Introduction

The American Kennel Club (AKC) is currently listing about 190 officially recognized dog breeds. Recognizing different dog breeds are important since different breeds of dogs have various characteristics and health conditions. However, it is hard for human beings to recognize all of them, especially when some dogs look similar but are of different breeds. Therefore, building a technical solution would help to solve the problem. This project uses transfer learning, which is an effective approach in machine learning. Three pre-trained deep network models, MobileNet v2, VGG16, and ResNet50, are loaded and applied to achieve the dog breed classification given an image. The models are modified based on the structure of the dataset and the needs of this project. In the continuation of this project, we compared the performance of the three models and did some experiments on them. The evaluation of the performance is mainly based on the accuracy rate, the average loss of each model, and whether the model tends to be overfitting or not. The computational speed is also taken into consideration to evaluate the efficiency of the models. Also, we undertook more experiments by modifying the parameters when training a model to see how the change of different aspects can affect the performance of the MobileNet model. To get a better understanding of how the model processes the data. We also analyzed the first layers of MobileNet and examined the weight of an image to demonstrate the result.

II. Related Work

Convolutional neural networks are particularly prevalent in different topics of deep learning, including but not limited to image recognition, detection and data generation. Three relevant papers on evaluating Stanford Dogs Datasets were further elaborated here. A study conducted by Raduly et al. [1] has focused on a fine-grained image recognition problem in identifying the breed of a dog in a given image. They employed two different convolutional neural network architectures: the NASNet-A mobile architecture and the Inception-ResNet-v2 deep architecture. Those architectural models used pre-trained models that were trained and tuned general recurring features already. The study tackled and mitigated overfitting issues by data augmentation using multiple transformations. The authors have also compared the results of two models on the selected dogs dataset. It turns out that the smaller mobile-friendly CNN model performed only 10% less accurately than the deep Inception-ResNet-v2 model.

Another research conducted by Lai et al. [2] that studied the outstanding problem of biometric identification of the dog specifically demonstrated fairly accurate and good results based on their trained results. Advanced machine learning techniques such as deep neural networks on pets photographs were utilized. The research explored "soft" biometrics such as breed, height and gender in contrast to "hard" biometrics as referenced by the authors such as pets' faces. Transfer learning principles were also applied in their research on the training of different CNN models. There were five periods in their training procedure. They started to employ three datasets in their work: Stanford, dogs dataset, Columbia dogs dataset and Flickr-dog dataset. Next they processed data using three stages: normalizing and resizing the images; splitting datasets into training and testing; final classifying breeds and recognizing identity using CNN classifiers. Then they trained an RCNN face detector based on VGG-16. The dog identification period was implemented in two individual stages. Firstly they conducted the "coarse stage" where they chose to examine InceptionV3, MobileNet, VGG-16 and Xception. Reasons being those networks are capable of classifying 1K object categories based upon the large scale of training. The research team incorporated the networks with the training done by ImageNet challenges. They have replaced the top-classification layers with an average pooling layer, two fully connected layers and a "softmax" layer. The preprocessing stage tuned the weights of extracted features from dog images. The final step was dog identification which the author referred to as the "fine stage". They narrowed down the datasets to only possible breeds. Data augmentation, identification that's find-tuned by transfer learning were implemented and repeated till all groups of dogs dataset were done testing. Their proposed network was able to successfully differentiate between two dog breed datasets with high accuracy of 90.8% and 91.29% when the so-called soft biometrics were utilized.

The third paper that's of great value to our project also used the Stanford dogs dataset. Kumar et al. [3] had utilized ImageNet to exercise fine-grained image cataloging. They divided the image into multiple lattices and a training batch size was set accordingly. They applied an algorithm to split and combine the descriptors and the channel information of the image which was extracted as the input of the convolutional neural network. CNN was used to extract the feature vectors from input images whereas ANN classified the input based on feature vectors generated. Their work configured ResNet architecture as the feature extractor and transfer learning for training. They used the softmax activation function as a classifier for prediction. The researchers proposed some future thoughts in increasing the performance of the model, such as employing a data masking on the training set to cancel the background noises especially noises from barcodes; applying dimensionality reduction techniques. Furthermore, instead of initializing the weights of the model trained on ImageNet, the model can be trained after initialization for better results.

III. Methods

The dataset used for this project is the Stanford Dog Dataset from Kaggles [4]. The dataset comprises 120 breeds of dogs. There are 10222 images in the dataset, and each image has a filename and a unique id, which is the breed of the dog.

Fig. 1: Sample images used in training models.

A. Packages Used

All the data processing and model training/experimenting steps described in this report were performed using the Pytorch, Pandas, NumPy, and Matplotlib packages. PyTorch is an open-source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing. Pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, and it is built on top of the Python programming language. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

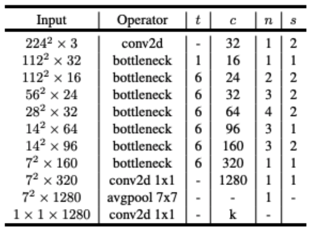
B. Data Preprocessing

The original dataset was split into training and testing datasets with 9222 images in the training set and 1000 images in the testing set. Then the original training and testing data were read into the program as data frames, and the strings of breed names were converted into numeric breed codes. A self-defined dataset was used. The dataset transforms all the images into a size of 224 * 224, converts the images to a tensor, and normalized the tensor using the mean and standard deviation values. After that, the datasets were transformed into data loaders with desired batch sizes. The data processing steps are completed and the data is ready for model training and testing.

C. Pretrained Models

Three pre-trained models are loaded from PyTorch and used in this project. Namely, MobileNet v2 [5], VGG16 [6], and ResNet50 [9]. To fit the data used in this project, the last layers of these three models are modified to have an output size of 120, which is the total number of unique labels in the dataset. The performance of the models is mainly evaluated on the accuracy and the average loss of the model. The computational speed of the models is also taken into consideration for the efficiency evaluation.

1) MobileNet-v2
MobileNet-v2 is a convolutional neural network(CNN) that is 53 layers deep. It was trained on more than a million images from the ImageNet database. Also, it is based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks that can have low latency for mobile and embedded devices. The biggest advantage of MobileNet is its high performance. It takes a much shorter time than the others, and its accuracy is impressive.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

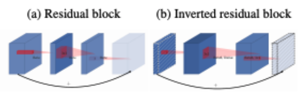


Fig. 2: MobileNet-v2 layers [5].

2) VGG16
VGG16 is also a convolutional neural network. It is an image classification neural network that uses 3*3 filters one after the another. Also, a batch normalization

layer is added to the VGG16 network [8] not only for overfitting but it provides the dataset batches one by one while the training period. The main concept of VGG16 is that it does not use high-level features making it a network for easy classification. It provides mainly all types of classification models. It mainly comprises two convolutional layers and followed by the pooling layers. These convo layers keep changing in their instances. As there are dense layers in the end the final output is carried by the dense layer which is then taken for validation or the accuracy level.

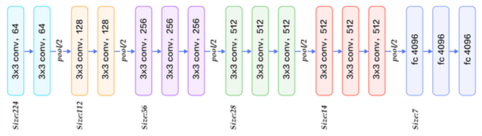


Fig. 3: VGG16 layers [9].

3) ResNet50
ResNet stands for Residual Network, which is an innovative neural network. ResNet50 is a variant of ResNet and it can work with 50 neural network layers. The ResNet50 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. The ResNet50 has over 23 million trainable parameters. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

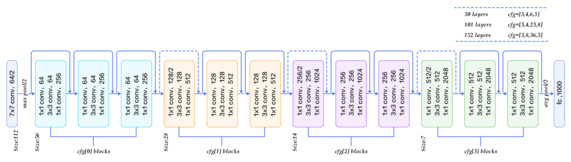


Fig. 4: ResNet50 layers [9].

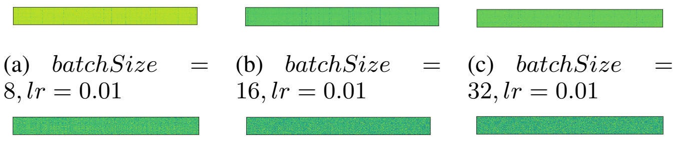
IV. Experiments and Results

A. Classifier Weights

We have examined our MobileNet model and analyzed its behavior in processing data. The weight of the linear classifier of the MobileNet model was accessed at two different learning rates (0.001, 0.01) and three different batch sizes (8, 16, and 32) were applied. The results were shown in the below Fig. 4.

B. Model Comparison

To compare and evaluate the performance of the three models, we calculated the computational speed, accuracy, and average loss of each model. With all the hyper-parameters remain the same(epoch size = 15, batch size = 64, learning rate = 0.001), Table 1 shows that in the training stage, all the three models achieve high accuracies and low losses. In terms of speed, mobilenet is the fastest.


(a) $batchSize = 8, lr = 0.01$ (b) $batchSize = 16, lr = 0.01$ (c) $batchSize = 32, lr = 0.01$

(d) $batchSize = 8, lr = 0.001$ (e) $batchSize = 16, lr = 0.001$ (f) $batchSize = 32, lr = 0.001$

Fig. 5: MobileNet classifier weights.

TABLE I: Model performance in the training stage.

	Speed(secs)	Accuracy	Avg loss
MobileNet v2	19465.7	96.4%	0.124946
Resnet	73551.7	97.5%	0.085238
Vgg16	29304.9	97.5%	0.099742

When applying the model to the test dataset with same configurations, VGG16 performs the best with an accuracy of 75.9% while the other two models only have accuracy rates of around 60%. Table 2 shows the details of the performances.

TABLE II: Model performance in the testing stage.

	Accuracy (train)	Accuracy (test)	Avg loss(train)	Avg loss(test)
MobileNet v2	96.4%	58.3%	0.124946	1.884860
Resnet	97.5%	57.5%	0.085238	1.843860
Vgg16	97.5%	75.9%	0.099742	0.885134

One possible reason for the gap between the training stage and the testing stage is model overfitting. Besides dogs, there are also backgrounds and other objects in the images. Therefore, the models may learn too many details and noises in the training data and fit too well. When it comes to the testing data, the details and noises can not apply or bring positive impacts. Another possible reason is that the size of the training epoch is not enough. The following plots shows that, for all the three models, the accuracies and average losses still tend to improve after the last epoch. Therefore, training for more epochs(maybe 30 - 50) may result in a better performance. However, due to the limitation of time and the current computational power of the machines, more epochs can not be achieved at this moment.

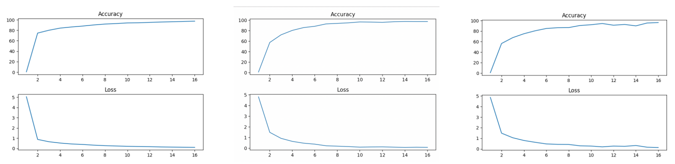


Fig. 6: Accuracy and average losses of each model(Mobilenet, ResNet, VGG16).

C. MobileNet Experiments

After comparing the result of different methods, we also tested the performance of MobileNet on different hyperparameters. Mobilenet was chosen for its faster computational speed.

The experiment was did in two dimensions, the training batch size, and learning rate. For batch size we chose 8, 16, and 32. For learning rate, we chose 0.001, 0.1, and 1. To save time, we reduced the training data size from 9000 to 5000, and instead of doing 15 epochs, we run 10 epochs for each experiment. The accuracies were lower than the previous training process, however, we can still get some insightful ideas from the trends of the models. Table 3 shows the accuracy for each experimental model, and Fig. 7 shows the trends of the accuracy and losses for each model.

TABLE III: Accuracy of each model.

Batch size / Learning rate	**0.001**	**0.1**	**1**
8	17.9%	1.0%	0.6%
16	71.2%	1.3%	1.1%
32	88.5%	1.3%	0.8%

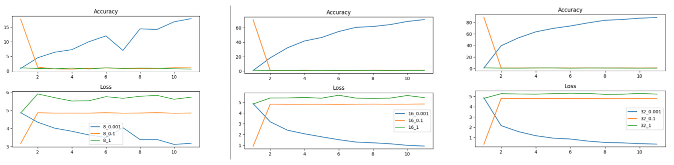


Fig. 7: Plots of accuracy and loss for each experimental models(batchSize = 8, 16, 32).

The table and plot show that a bigger batch size lead to a higher accuracy. For the learning rate, that the model is improving only with the learning rate 0.001. Larger learning rates do not work for this dataset.

V. Discussion and summary

We also tried to implement an original CNN network by ourselves, it is a 4-layer CNN model, we tried different basic methods and tuned the hyperparameters, but the result is always very bad, the accuracy is lower than 5%, which means the network doesn't work at all. We think it is frustrating but make sense because the pre-trained models that we found are all much more complex and every hyperparameters are all fine-tuned. Another reason for this low accuracy could be because the traning dataset is not big enough, as we mentioned above, the pre-trained models using ImageNet which includes already many dog breed data. The dataset we use is too small for training an original CNN network.

Ideas and Findings

From our exploration of different pre-trained models and self-made model, we found that the tuning of model is very important, even for the same model, with inappropriate hyperparameters the result is unbelievably bad. For example when we tried the vgg16, which was quite famous and was expected to have a good performance, we tried with the same hyperparamter as the mobilenet that we used before, but the accuracy was only 4.7%, and the average loss was 4.7. After doing some research and conducting some experiments, we improved the accuracy on training dataset to 97.5%, and the average loss was under 0.01. For the other two models we did the same thing, after many trials the result is much better. We figure that for different models their hyperparameters are quite different, which means we can't simply reuse the settings from other models, their design and features requires us to dive deep into the model and make relevant adjustments.

Also, the performance of a model may be influenced by the dataset. One problem in this project is that the images may have noises, which lead to a over training of the models. To solve this problem, we can do an image segmentation to detect the dog object before training the model. This can eliminate the influences of backgrounds or other noises.

Besides, we found that there is a trade-off between time and accuracy. For example, the MobileNet v2 is the fastest among the three models we used, it has less layers and takes much shorter time to train. However, its training accuracy is relatively low in comparison with the other 2(of course 96.4% itself is really good). It can be seen as a balance between performance and time spent. The self-trained model, which has the worse result, is much faster than the rest, it takes only $\frac{1}{3}$ time of the MobileNet v2 because its far less complex. We found that there is no best model but the most suitable model in solving real world problems. On customer side, sometimes the 95% and 96% doesn't make much difference while using the model, but the speed can vary from 0.1 secs to 2 secs, which highly influence the customer experience. On developer side, the speed of the model can influence how many resourses(like GPUs) that need to be used to train models, which can save money and time. Therefore, we think the accuarcy is not the only standard for choosing model, we also need to take into consideration the specific real-world implementation and based on our requirements to choose the most suitable model.

References

[1] Z. Ráduly, C. Sulyok, Z. Vadászi and A. Zölde, "Dog Breed Identification Using Deep Learning," 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), 2018, pp. 000271-000276, doi: 10.1109/SISY.2018.8524715.

[2] K. Lai, X. Tu and S. Yanushkevich, "Dog Identification using Soft Biometrics and Neural Networks," 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8851971.

[3] R. Kumar, M. Sharma, K. Dhawale and G. Singal, "Identification of Dog Breeds Using Deep Learning," 2019 IEEE 9th International Conference on Advanced Computing (IACC), 2019, pp. 193-198, doi: 10.1109/IACC48062.2019.8971604.

[4] Stanford Dog Dataset from Kaggles: https://www.kaggle.com/competitions/dog-breed-identification/data.

[5] MobileNet v2 Model: https://pytorch.org/hub/pytorch_vision_mobilenet_v2/.

[6] VGG16 Model: https: //pytorch.org/vision/main/generated/torchvision.models.vgg16.html.

[7] ResNet50 Model: https://pytorch.org/hub/nvidia_deeplearningexamples_resnet50/.

[8] VGG16 Layers: https://iq.opengenus.org/vgg16/.

[9] ResNet50 Layers: https://blog.devgenius.io/resnet50-6b42934db431.