Dog Breed Classification using Fine-tuned pretrained Models and Ensemble Models

Bowen Liang University at Buffalo bowenlia@buffalo.edu bowenlia	Zihan Wang University at Buffalo zwang223@buffalo.edu zwang223	Guanchong Huang University at Buffalo guanchon@buffalo.edu guanchon

1 Introduction

Convolutional neural network (CNN) is a classical model in machine learning. Its expanded network and expanded function play an important role in image classification and recognition. In the research of CNN, due to the complexity of its parameters, it often requires a lot of time and energy to adjust the parameters to achieve better output results. In order to realize dog breed recognition, this project first built the CNN from scratch framework and adjusted the parameter structure. According to the performance of the experimental results, we used three pretrained models: ***VGG16***, ***ResNET50*** and ***InceptionV3***, and in the practice stage of these three models, we have achieved good accuracy after adjusting the hyperparameters. In the last stage of the experiment, the team members fused the features of the three models used into an *ensemble model* and significantly improved the accuracy. Finally, we evaluated the accuracy, advantages and disadvantages of the above methods and proposed future improvement.

Keyword: CNN, Dog Breed, Pretrained, Ensemble model

Repository：https://github.com/NicholasL4/Dog-Breed-Classification-using-Ensemble-Pretrained-Models

2 Background

2.1 Why CNN

The concept of CNN is very important in network technology. With the advent of 5G era, the transmission of pictures and videos will be faster and faster, CNN's resolution and recognition ability plays a very important role. This network technology is often used to recognize faces and objects. If it is extended to more specific recognition content, it can also be used to identify human's closest partner, pet cats and dogs. Taking pet dogs as an example, people like their companionship, but it is often difficult to distinguish dog breeds. Applying image recognition to dog breeds can help people solve daily little puzzles, which is also a meaningful and human function. Of course, this application has a very *broad prospect*. We can obtain the appearance information of pets when designing the model. On the basis of classification, we can bind the appearance characteristics of dogs with their owner identity, so as to provide strong help for pet registration, pet safety protection, pet medical treatment, and pet lost/found.

2.2 Implement the model

Convolution and pooling have greatly improved the accuracy of CNN recognition. At the same time, due to the complexity of its parameters, the practice process is still full of challenges. This network

structure depends on the capacity, integrity, and accuracy of the data set. Our project uses *Stanford's dog data set*, which contains 120 dog breeds, and each breed has hundreds of corresponding pictures, which is enough for in-depth learning testing and practice. However, in practice, we will still have some unsatisfactory results, which is also a common problem in CNN, such as over fitting and low recognition accuracy. In this regard, team members believe that the accuracy can be improved by using pre trained models or adding training data. *Pre-training + fine tuning* is a very popular trick in deep learning, especially represented by the image field. In many cases, the pre-trained ImageNet will be selected to initialize the model [1].

Transfer learning can help us use pretrained models to improve the accuracy of results, while reduce training time and computing resources. We adjusted the parameters of the pretrained models to expect higher accuracy results. This project uses CNN from scratch and three pretrained models: ***VGG16***, ***ResNET50*** and ***InceptionV3*** deep learning methods to deal with dog recognition and uses the reusable model to improve the accuracy.

3 Dataset and Features

This project uses the *Kaggle Competition Dataset* contained in the Stanford dog dataset [2], which is constructed using images and annotations in ImageNet for fine-grained image classification tasks. A total of 10222 images were used in the experiment. Each image belongs to one of the 120 dog breeds, that is, there are 120 subfolders under the images folder, and each subfolder only contains the pictures of dogs of the same breed. We use 80% of the images as the training set and 20% of the images as the validation set for cross validation. At the same time, we use the method in ***Keras*** for data preprocessing on the basis of data acquisition.

4 Method

4.1 CNN from scratch

First, we used CNN from scratch with some tuning to see how accurate it can be and then further comparing the result with other pretrained models and ensemble models.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_10 (Bat  (None, 224, 224, 3)       12
chNormalization)

conv2d_36 (Conv2D)           (None, 222, 222, 16)      448

max_pooling2d_32 (MaxPoolin  (None, 111, 111, 16)      0
g2D)

batch_normalization_11 (Bat  (None, 111, 111, 16)      64
chNormalization)

conv2d_37 (Conv2D)           (None, 109, 109, 32)      4640

max_pooling2d_33 (MaxPoolin  (None, 54, 54, 32)        0
g2D)

batch_normalization_12 (Bat  (None, 54, 54, 32)        128
chNormalization)
```

Different filters and different output structures in CNN(Part.1)

```
conv2d_38 (Conv2D)           (None, 52, 52, 64)        18496

max_pooling2d_34 (MaxPoolin  (None, 26, 26, 64)        0
g2D)

batch_normalization_13 (Bat  (None, 26, 26, 64)        256
chNormalization)

conv2d_39 (Conv2D)           (None, 24, 24, 128)       73856

max_pooling2d_35 (MaxPoolin  (None, 12, 12, 128)       0
g2D)

batch_normalization_14 (Bat  (None, 12, 12, 128)       512
chNormalization)

conv2d_40 (Conv2D)           (None, 10, 10, 256)       295168

max_pooling2d_36 (MaxPoolin  (None, 5, 5, 256)         0
g2D)

global_average_pooling2d (G  (None, 256)               0
lobalAveragePooling2D)

dense_18 (Dense)             (None, 120)               30840

=================================================================
Total params: 424,420
Trainable params: 423,934
Non-trainable params: 486
_________________________________________________________________
```

Different filters and different output structures in CNN(Part.2)

4.2 VGG16

VGG was proposed by the visual geometry group of Oxford[4]. The network proves that increasing the depth of the network can affect the final performance of the network to a certain extent. An improvement of VGG16 over AlexNet is to use several consecutive 3*3 convolution kernels to replace the larger convolution kernels (11*11, 7*7, 5*5) in AlexNet. For a given receptive field (the local size of the input picture related to the output), the use of stacked small convolution kernel is better than the use of large convolution kernel, because multi-layer nonlinear layers can increase the network depth to ensure the learning of more complex patterns, and the cost is relatively small. In brief, in VGG, three 3*3 convolution cores are used to replace the 7*7 convolution core and two 3*3 convolution cores are used to replace the 5*5 convolution core. The main purpose of this is to improve the depth of the network and the effect of neural network to a certain extent under the condition of ensuring the same perception field.

VGG16 contains 16 hidden layers (13 convolution layers and 3 full connection layers). The structure of VGG16 is very simple. The whole network uses the same convolution core size (3*3) and maximum pool size (2*2). Its structure also verifies that the performance can be improved by deepening the network structure. However, VGG consumes more computing resources and uses more parameters, resulting in more memory consumption, most of the parameters are from the first full connection layer [4].

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688

batch_normalization_12 (Bat  (None, 7, 7, 512)         2048
chNormalization)

global_average_pooling2d_14  (None, 512)               0
 (GlobalAveragePooling2D)

dropout_36 (Dropout)         (None, 512)               0

dense_42 (Dense)             (None, 1024)              525312

dropout_37 (Dropout)         (None, 1024)              0

dense_43 (Dense)             (None, 256)               262400

dropout_38 (Dropout)         (None, 256)               0

dense_44 (Dense)             (None, 120)               30840

=================================================================
Total params: 15,535,288
Trainable params: 819,576
Non-trainable params: 14,715,712
_________________________________________________________________
```

VGG16 model summary

Batch Normalization and ***GlobalAveragePooling2d*** methods are used in the application of VGG in this project. Batch normalization, similar to ordinary data standardization, is a way to unify scattered data and a method to optimize neural network, the normalization has unified data, which makes it easier for machine learning to learn the laws in the data [5].GlobalAveragePooling2d simplifies a large number of parameter operations.

The first two of the three full connection layers use ***Relu***, and the last one uses ***Softmax*** as the activation function. Finally, while using ***Adam Optimization*** method to optimize and locate the learning rate at 0.001, we also try ***RMSprop Optimization*** method.

VGG16 model process

4.3 ResNet 50

In Residual Nets, the design of deep residual network is to overcome the problems of low learning efficiency and ineffective improvement of accuracy caused by the deepening of network depth[6].

Resnet50 has two basic blocks, named *Conv Block* and *Identity Block* respectively. The input and output dimensions of Conv Block are different, so they cannot be connected in series continuously. Its function is to change the dimension of the network. The input dimension and output dimension of Identity Block are the same and can be connected in series to deepen the network. We also tuning the output by applying ***Batch Normalization, GlobalAveragePooling*** and ***Relu*** layers.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712

batch_normalization_2 (Batc  (None, 7, 7, 2048)        8192
hNormalization)

global_average_pooling2d_2   (None, 2048)              0
(GlobalAveragePooling2D)

dropout_4 (Dropout)          (None, 2048)              0

dense_4 (Dense)              (None, 2048)              4196352

dropout_5 (Dropout)          (None, 2048)              0

dense_5 (Dense)              (None, 120)               245880

=================================================================
Total params: 28,038,136
Trainable params: 4,446,328
Non-trainable params: 23,591,808
_________________________________________________________________
```

ResNet50 model summary

4.4 InceptionV3

The **main idea** of inception architecture is to find out how to approximate the optimal local sparse junction with dense components. InceptionV2 draws on the design idea of VGGNet and replaces the 5*5 filter with two 3*3 filters. InceptionV3 mainly improves concept V2 in two aspects. First, V3 optimizes the structure of the inception module. Now there are more types of inception modules, and inception V3 also uses a branch in the branch of inception module. Secondly, the method of splitting a larger two-dimensional filter into two smaller one-dimensional filters is also introduced in V3[7].

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784

batch_normalization_189 (Ba  (None, 5, 5, 2048)        8192
tchNormalization)

global_average_pooling2d_1   (None, 2048)              0
(GlobalAveragePooling2D)

dropout_2 (Dropout)          (None, 2048)              0

dense_2 (Dense)              (None, 1024)              2098176

dropout_3 (Dropout)          (None, 1024)              0

dense_3 (Dense)              (None, 120)               123000

=================================================================
Total params: 24,032,152
Trainable params: 2,225,272
Non-trainable params: 21,806,880
```

InceptionV3 model summary

4.5 Ensemble model using averaging

In machine learning, ensemble models combine decisions from multiple models to improve the overall performance, which is **the focus of this project**. We used two different approaches to build ensemble model the first one is averaging and another one is stacking.

We ensembled the network from previously tuned ResNet50, InceptionV3, and VGG16. *Average* uses multiple models to predict each data point. The prediction of each model is regarded as "voting", and each data point in the average is predicted multiple times. In this method, we obtain the average value of the prediction from all models and use it to make the final prediction as well as the output with different weights for each model.

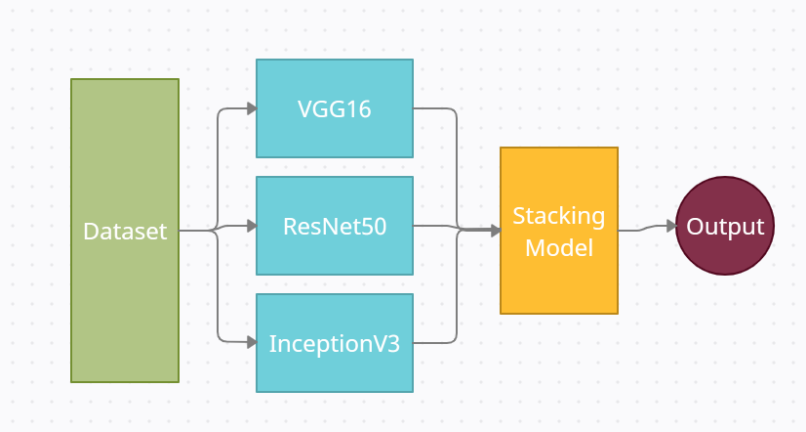
```
Model: "model_13"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================================
 input_39 (InputLayer)       [(None, 224, 224, 3       0           []
                             )]

 sequential_21 (Sequential)  (None, 120)               15535288    ['input_39[0][0]']

 sequential_22 (Sequential)  (None, 120)               25817080    ['input_39[0][0]']

 sequential_23 (Sequential)  (None, 120)               24032152    ['input_39[0][0]']

 average_5 (Average)         (None, 120)               0           ['sequential_21[5][0]',
                                                                    'sequential_22[5][0]',
                                                                    'sequential_23[5][0]']

==================================================================================================
Total params: 65,384,520
Trainable params: 5,270,120
Non-trainable params: 60,114,400
__________________________________________________________________________________________________
```

Ensemble model using averaging model summary

4.6 Ensemble model using stacking

Ensemble model using stacking, as an integrated learning technology, it uses predictions from multiple models (such as Decision Tree, KNN or SVM) to build new models [8].

The model is ensembled using the networks we tuned previously in ResNet50, InceptionV3, and VGG16. The detailed setup of the ensemble model is further illustrated on the figure below. By stacking these 3 models it should outputs the dog breed classification more accurately.



Stacking architecture

5 Experiments and Results

5.1 GitHub Repository

The detailed loss and accuracy graphs for each model are contained in *ipynb* files within the repository.
https://github.com/NicholasL4/Dog-Breed-Classification-using-Ensemble-Pretrained-Models

5.2 Results

Model	Training accuracy	Validation accuracy
CNN from scratch	98.47%	19.25%
VGG16	51.60%	65.36%
ResNet50	94.14%	75.10%
InceptionV3	90.77%	81.16%
Ensemble using Averaging	75.95%	67.17%
Ensemble using Stacking	80.86%	91.15%

After tuning the hyperparameter several times we got the results above. As we can see that the validation accuracy from *CNN from scratch* is very low compared to other pretrained models as expected. However, the validation accuracies from VGG15, ResNet50 and InceptionV3 are not as well as expected. One explanation is that the size of the dataset is very limited and deep learning requires a lot of data-driven process. Since the dataset only contains 11,000, it may impact the validation accuracy. [9]

The two different approaches from averaging and stacking give different results. As we can see, averaging method only returns a better accuracy than VGG16 by 2% while stacking method gives the best overall accuracy at 91.15% and it is higher than the results achieved by each of the models individually.

5 Conclusions

In this project, we implemented and tuned several models and methods to obtain dog breed classification, and further implemented ensemble models with ImageNet weights to get better accuracy. We found that the CNN from scratch didn't do a well job because of not enough data and maybe there are not enough hidden layers. In some pretrained models, like VGG16, ResNet50, and InceptionV3, they showed some high accuracy in the results and InceptionV3 gives the best result among these three models. Finally, the ensemble models using stacking improved the accuracy a lot by 10%.

Reference

[1] zhyuxie, Transfer learning, 01/06/2019. https://blog.csdn.net/dakenz/article/details/85954548

[2] Aditya Khosla,Nityananda Jayadevaprakash,Bangpeng Yao,Li Fei-Fei, StanfordDogsDataset http://vision.stanford.edu/aditya86/ImageNetDogs/

[3] Clay Mason, Dog Breed Image Classification, 12/12/2018. https://medium.com/@claymason313/dog-breed-image-classification-1ef7dc1b1967

[4] Amusi, Read VGG network, 08/06/2018. https://zhuanlan.zhihu.com/p/41423739

[5] mofan, Why need batch normalizaiton?, 05/12/2017. https://zhuanlan.zhihu.com/p/24810318

[6] aixiuxiandedaoyou, Classical convolutional network of ResNet50, 03/12/2020 http://jianshu.com/p/c3ddc5a5e8b6

[7] xiaojingyunketang, Classical convolutional network of InceptionNet-V3, 09/23/2019. https://zhuanlan.zhihu.com/p/83807421

[8] Aishwarya Singh, A Comprehensive Guide to Ensemble Learning (with Python codes), 06/18/2018. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

[9] T. Linjordet andK. Balog, "Impact of Training Dataset Size on Neural Answer" in Advances in Information Retrieval, vol. 11427, pp. 828-835, 2019.