Transfer Learning for Dog Breed Identification

Yizhou Chen yic244@ucsd.edu
Zhongke Ma z2ma@ucsd.edu
Qichao Zheng q5zheng@ucsd.edu
Yisheng Ji y3ji244@ucsd.edu

Abstract

In this project, we achieve classification of dog breeds from Stanford Dog Dataset which contains images of 120 dog breeds. ResNet50, DenseNet, VGG19 and InceptionV3 were used to extract the features. To get a better accuracy, we apply transfer learning on different networks such as VGG19 and InceptionV3. These networks were already built on Keras, which is a deep learning library and provides some popular neural networks. The result of this project can be used in identification of dogs in images.

1 Introduction

1.1 Motivation

Dog have become more and more important in our daily life, they appear as pets, transportation, drug detector and so on. Since different breed of dogs can help people in different way, the identification on breed of dogs have become commonly mentioned nowadays in many fields. Thus, we need a more accurate and efficiency way to classify the breed of dogs.

1.2 Challenges

1.3 Convolutional Neural Network

Deep learning models have achieved remarkable results in computer vision in recent years. [1] As one of the deep learning networks, Convolutional Neural Network has widely used around the world for image classification. The CNN consist of neurons and the weights and biases of the neurons can be trained. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. Convolutional layer, pooling layer, and fully-connected layer are the three main types of layers of Convolutional Neural Network.

In order to improve the performance of the network, CNN model nowadays has become deeper. However, as CNN become deeper, the cost of time and computing resource also become larger. Thus, to save time and make our calculation more efficient, we will use a pre-trained model in our project. Pre-trained model is a model already trained for some other task, we can re-train them in order to fit our task.

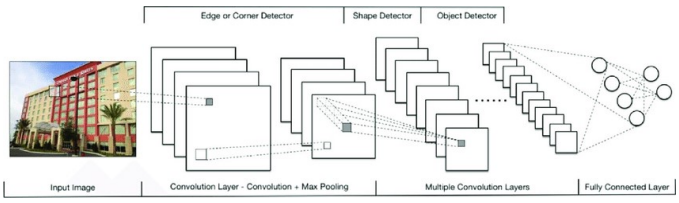

Figure 1: Basic structure of CNN.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Description of methods

In this project, we will adopt several CNN models for the purpose of image classification. The past ten years have witnessed great success in the performance of deep network handling the problem of image classification. And there have been serverl great framewark or architectures successful achieve below the 5% top five error rate. Such as the **DenseNet**, **ResNet** and **Inception**. The former two methods adopt the concepts of *shortcut connections*, while the latter one utilize the manner of *factorizing convolutions*. The purpose of these methods is inituitive, by reducing the parameter of deep network so as to allow for the network to grow deeper and deeper.

Therefore, in this project, we wish to explore the difference between these methods. Namely, we wish to see the difference of *a) shortcut connections* and *b) factorizing convolutions* in perforing large scale, multi-labeled image classifications. Unlike the basic classifcation problems, where the objects being classified varied in a wide range. In this projects, we wish to examine how well the deep network can handle the classification of similar objects. Given thst thoes target objects of the same catagory can somehow varied in a wode range.

Further, we compare the performance of **DenseNet** and **Inception V3**, together with a earlier framework **VGG19**, to examine the performance under different deep network.

2.1 VGG19

VGG is a network of increasing depth using an architecture with very small convolution filters. With small convolution filters, the number of parameters can be reduced. The input of VGG is 224*224 RGB images. Then the input image will go through the convolutional layers, where filters with a small receptive field will be applied. Each convolutional layer has different depth, some of the convolutional layers will have a max-pooling layer followed, and the max-pooling layer has a 2×2 pixels window with stride 2. With max-pooling layers, VGG can achieve spatial pooling. After the stack of convolutional layers three fully connected layers are followed. The first two layers have 4096 channels each, the third have 1000 channels. And the last layer of the network is a soft max layer. After the input go through the whole network, the output will be classification contains 1000 channels. VGG16 and VGG19 are the two most common models of VGG. VGG 16 has 16 weight layers which has about 138 million of parameters in total, while VGG19 has 19 weight layers which has about 144 million of parameters in total. [4] The model we will use in our project is VGG19, it can achieve 7.3% top 5 error in ILSVRC.

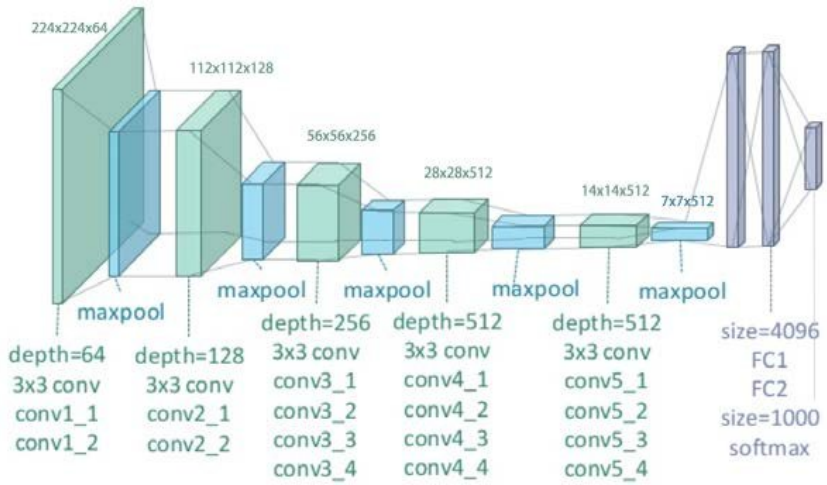

Figure 2: VGG19 architecture.

2.2 DenseNet

More and more models have shown that convolutional neural networks can become substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close

to the input and those close to the output. Thus, Dense Convolutional Networks (DenseNet) was introduced. DenseNet consists of dense block, within each dense block, each layer is connected to every other layer in a feed-forward fashion. [3] In DenseNet, every layer is connected together by dense connectivity in order to have max information flow between the layers which means every layer have multiple inputs from other layers.

In DenseNet, fewer parameters is needed, which means the parameters in DenseNet can be more efficiency than traditional neural networks. Besides, DenseNet can be deep while the layers can be slim. With all these advantages, DenseNet is easily to be trained.

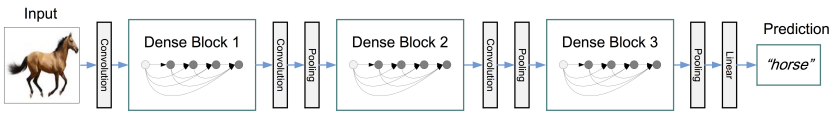

Figure 3: A deep DenseNet with three dense blocks.

2.3 InceptionV3

Inception is a deep convolutional neural network which provides a new state of the art for classification and detection. It makes a better use of the computing resources inside the network and makes the network deeper and wider while the computing budget is not affected. Inception networks consist of Inception modules. The main idea of the Inception architecture is to consider how an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components. [5] To find the optimal local sparse structure, we assume that each unit of the previous layer refers to some part of the input image, which can find the clusters that focused on some particular region, thus, these clusters can be summed up by a 1*1 convolution in the following layer. And to avoid patch-alignment problem, the filter size of Inception has only 3 options: 1×1, 3×3 and 5×5. To keep the computing complexity stable, Inception network uses dimensionality reduction which can reduce the feature maps from 192 to 16.

InceptionV3, as the improved model of Inception networks, has 42 layers, which is much deeper than GoogLeNet, the first model of Inception networks. The improvement InceptionV3 make is that they applied factorization of convolutions and improved normalization in the networks. [6] Comparing with GoogLeNet, the top 5 error of InceptionV3 is improved from 6.67% to 4.49%.

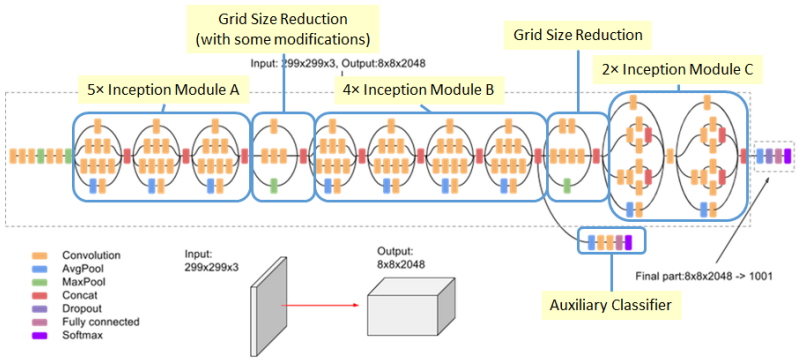

Figure 4: InceptionV3 architecture.

2.4 Transfer learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. In traditional CNN for image classification, usually the earlier layers are to detect edges, while deeper layers will try to detect some specific features. With transfer learning, we can use the earlier layers to re-train the deeper layers. The main benefit transfer learning is saving

training time without hurting the performance. Also, with transfer learning, we can use less data to train the neural networks.

3 Implementation Details

In this section, we will brefily introcude how we implemented and modified these three deep network mentioned above.

3.1 Models

Basically, we first examine the performance of these three pretrained network on our testing set; then we remove the final layers (top layers) within the pretained models, plugged a new fully-connected layers as the new classifer. During training, we first forzen the parameters of previous layers, since they all have been properly trained and fine-tuned, and trend to achieve optimal performance on general classification problems. These previous layers, then functioned as feature extarctor, extracting the features from the input image tensors, generating a group of feature vectors, passing through the last fully connected layer, and fire on some specfied output nodes indicating the predicted catagories of the input images.

Table 1 demenstartes one of the prototype of the deep network that will be adopted in our project.

Table 1: A typical modified framework for our project.

layer	function	input dims	output dims
0	*input images*	$(n, 3, 224, 224)$	–
$\cdots$	–	–	–
$n-1$	*extract deep features*	–	$(n, n, n, 2048)$
n	*global average pooling*	$(n, n, n, 2048)$	$(n, 2048)$
$n+1$	*drop out*	$(n, 2048)$	$(n, 2048)$
$n+2$	*fully connected*	$(n, 2048)$	$(n, 120)$

3.2 Dataset

As for the dataset, we have $12,000$ samples for training and $8,580$ samples for testing. We further adopted the method of data augmentation to enhance the numbers of the training set. More specifily, we adopt some random *shifts*, *zooms*, *rotates* and *filps* to tarnsform the origin image, so as to obtained the augmented data for training. This method, does somehow improve the performance of our model.

3.3 Losses

As for the loss function, we basically use both the *categorical_crossentropy* and *sparse_categorical_crossentropy* that are typically adopted in classification problem. Also, by some literature review, we have also try to adopted the *center loss* $\mathcal{L}_C$, as a regualrized term for the training of our model, which is found to be significant useful in classifying similar targets. Therefore, we define the loss of our model as

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C \tag{1}$$

where $\mathcal{L}_S$ is the softmax loss and $\mathcal{L}_C$ is the center loss defined by

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W^T x_i + b_i}}{\sum_{j=1}^{n} e^{W^T x_j + b_j}} \tag{2}$$

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^{m} \|x_i - c_i\|_2^2 \tag{3}$$

4 Experimental Settings

4.1 Datasets

The dataset we used in this project is Stanford Dogs Breed Dataset, which contains images of 120 breeds of dogs from around the world, such as Chihuahua, Japanese spaniel and Rhodesian ridgeback. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. There are totally $20,580$ images in this dataset, with $12,000$ images in the training set and $8,580$ in the testing set.

For training-validation split rate, wo use $1/6$ and introduce data augmentation methods including left/right shifting ± 0.2, scaling 1 ± 0.2, rotation ± 30 and horizontal flip. The training, testing and validation data in a batch scale of 32 samples per batch, and in total we have 315 batches for training, 60 batches for validation and 269 batches for testing. Figure 5 shows 10 sample images from the dataset with class label of *Border Collie*.

Figure 5: Sample Images from the Stanford Dogs Breed Dataset. Catagory =*Border Collie*

4.2 Training Parameters

Undoubtly, the most important and most difficult part of trainning a deep network, is how to properly choose the *1)* layers, *2)* losses and *3)* optimizers. All these staffs are characterized by parameters and closely related with the performance of the whole network. Table 2 demenstartes some of the parameters that we adopted for training the plugged layers.

Table 2: A typical modified framework for our project.

plugged layer	activations	optimizeor	step / epoch	loss	early stopping
fc64, fc120	ReLu, Softmax	adam + sgd	120	$\mathcal{L}_S$	*true*
d0.2, fc120	–, Softmax	sgd	80	$\mathcal{L}_S$	*true*
d0.4, fc120$\times$2	–, Softmax	adam + sgd	120	$\mathcal{L}_S$	*true*
fc64, fc120	GLU, Softmax	adam + sgd	120	$\mathcal{L}_S + \mathcal{L}_C$	*true*

As for finetuning, we adopt a two-step finetune manner. Namely, we fine-tune the network for two stages, while in the first stage, we use the less augmented data, with only left/right shifting ± 0.1 and scaling 1 ± 0.05. And for the second stage, we use the non-augmented data, simply set all augment manner to *false*. As for the optimizer, we again, utilize sgd, with a learning rate of 0.0001 and momentum of 0.9. For each stage, we run for 200 epochs with a step size of 120 batches per epoch, toger ther with early stopping if the validation loss start to increase.

4.3 Hardware & Framework

For the whole project, we use the *Keras* along with the *Tensorflow* backend, running on both the UCSD's MLDSP Clusters equipped Ndivda GeForce 1080Ti and our both PC with GeForce 1060. To avoid the problem of packages' version compatibility, we adopted the offical *tensorflow*'s docker.

5 Results

5.1 InceptionV3

5.2 DenseNet121

5.3 VGG19

Number of selected categories	Accuracy
20	87.31
40	83.53
60	80.01
80	76.29
100	74.42
120	70.67

Table 3: Accuracy versus categories selected

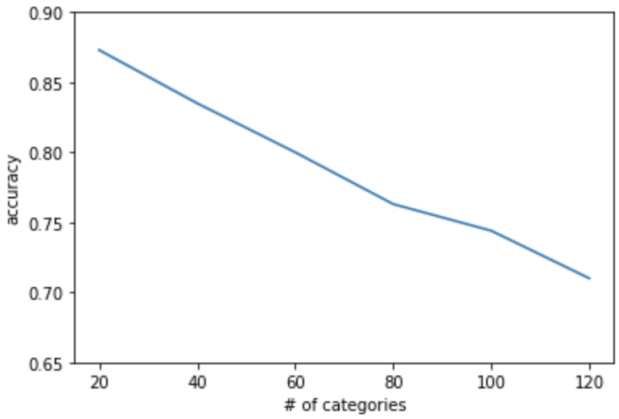

Figure 6: Accuracy versus categories selected

The test accuracy of our algorithm is 70.07% if all 120 dog breeds are included.

5.4 Comparision

6 Discussion

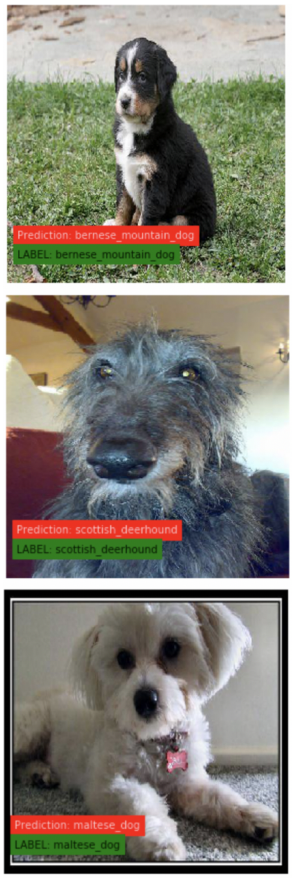

Figure 7: Some correctly classified samples

Figure 8: Some misclassified samples

From the above misclassified samples, the color of the background is so much close to the dog, which will We also notice that in the training test there exist some images contains more than one dog and same image appears twice in two different breeds as shown in below. This type of error seems not solvable via programming.

Figure 9: Sample of more than one dogs

7 Appendix

References

[1] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[2] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[3] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//CVPR. 2017, 1(2): 3.

[4] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[5] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[6] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.