Capstone Project

Machine Learning Engineer Nanodegree

Liang Huiying
December 7, 2016

I. Definition

Project Overview

Cats and dogs are the most favorite pets in a human's life, therefore, we always have photos containing cats or dogs in our albums. Sometimes we try to classify these photos into categories. What if we can do these things automatically rather than by our own hands or eyes? With the technology of computer vision, the knowledge of machine learning or more precisely deep learning, these problems can be solved.

Many people have tried different ways to solve this problem and submitted their results on kaggle[1]. Most of them really did very good jobs while some are not so satisfied.

In this project, I built a classifier trained by the dataset downloaded from kaggle. The dataset contains a large amount of ordinary daily images including cats or dogs. The classifier runs on the Jupyter Notebook.

Project Statement

This is a supervised learning[2] problem because each example in training data is a pair consisting of an input image and a desired output value (*cat* or *dog*). The classifier implements a binary classification considering the goal is to tell you if it is cats or dogs in the image when you put an image into the classifier.

The project consists of the following parts:

1 Download dataset from kaggle.
2 Preprocess the data.
  2.1 Divide images in *train* folder into a training set and a validation set.
  2.2 Resize and rescale images in *train* folder.
3 Build the structure of the classifier.
  3.1 The basic structure of the classifier is ResNet-50.
  3.2 Transfer learning is used to improve the training speed and accuracy.
4 Train the classifier.
5 Use the classifier to recognize cats and dogs.

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
[2] https://en.wikipedia.org/wiki/Supervised_learning

Metrics

Accuracy, time and log loss[3] are used to evaluate the performance of the classifier. Accuracy and log loss are typical metrics for a classifier.

Accuracy will show how well the classifier performance, what percentage of the result is true. The higher the accuracy, the better the classifier.

$$accuracy = \frac{n}{N} * 100\%$$

- n: number of right predictions of dogs or cats
- N: total number of images in test dataset

The use of log on the error provides extreme punishments for being both confident and wrong. When the prediction is definitely true, log loss equals 0. When the prediction is definitely false, log loss will be extremely big. A smaller log loss means a better classifier.

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(\widehat{y_i}) + (1 - y_i)\log(1 - y_i)]$$

- n: the number of images in the test set
- $\widehat{y_i}$: the predicted probability of the image being a dog
- y_i : 1 if the image is a dog, 0 if cat

Time is also a useful metrics for this classifier, it could tell how fast the classifier runs.

$$time\ = \frac{T}{N}$$

- T: time used for predicting the test dataset
- N: total number of images in test dataset

II. Analysis

Data Exploration

The dataset downloaded from kaggle contains two files, *test* and *train*. The *train* folder contains 12500 images of dogs and 12500 images of cats. Each image in this folder has the label as part of the filename. The *test* folder contains 12,500 images, named according to a numeric id without label. Images from *test* or *train* folders are ordinary daily images including cats or dogs with shapes and sizes.

[3] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/details/evaluation

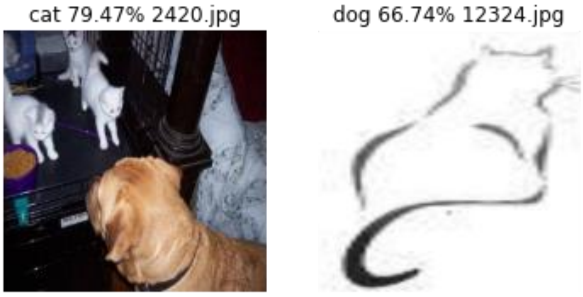

Figure.1. Images from dataset

Figure.1. shows two of the images which are difficult to predict. The left image contains both cats and dogs, so it is hard for the classifier to say it is cat or it is dog. The right image is too abstract and most of images in training dataset is photos from our daily life.

I split *train* folder into 2 folders:

- ❖ *mytrain* ---- including two folders
  - ➢ *cat* ---- including about 11250 cat images
  - ➢ *dog* ---- including about 11250 dog images
- ❖ *myvalid* ---- including two folders
  - ➢ *cat* ---- including about 1250 cat images
  - ➢ *dog* ---- including about 1250 dog images

Exploration Visualization

Figure.2. Images from dataset

Figure.3. Images from dataset

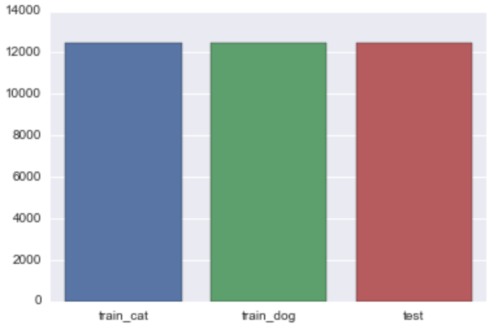

Figure.4. number of images from original dataset

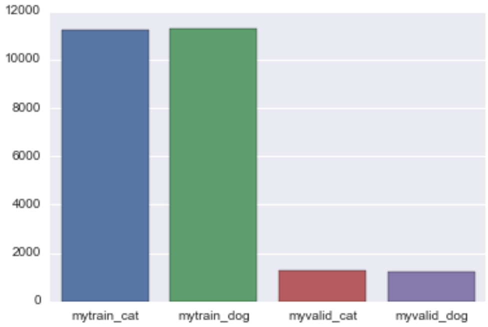

Figure.5. numbers of images after splitted

Figure.4. shows the original structure of the dataset. 12500 cat images and 12500 dog images in the training dataset and 12500 images contains cats or dogs in total in the test dataset. It is a balanced dataset which is good for the classifier to learn.

Figure.5. shows numbers of images in train dataset after splitted in *mytrain* and *myvalid*. *Mytrain* contains about 11250 cat images and 11250 dog images and *myvalid* contains about 1250 cat images and 1250 dog images.

Algorithms and Techniques

The basic structure of the classifier is ResNet-50[4]. The full name of ResNet is Deep Residual Networks. 50 means this model contains 50 layers. ResNet was invented by Kaiming He and his team for the reason of image recognition and won the 1st places in ImageNet classification. ImageNet is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images[5].

ResNet is one of the Convolutional Neural Networks(CNNs) [6]. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume. Instead, the convolutional layers connect each neuron to only a local region of the input volume which shows in Figure.6.

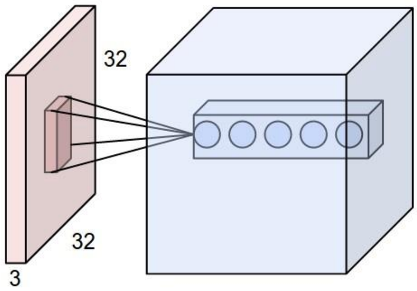

Figure.6.

Parameter sharing scheme is used in convolutional layers to control the number of parameters. Because of parameter sharing, convolutional neuron networks are not sensitive about the positon of targets.

Convolution is a mathematical operation[7]. In a convolutional neuron networks, convolution is used to computing the feature map of an image. The computing process shows in Figure.7. and Figure.8.

[4] https://github.com/KaimingHe/deep-residual-networks

[5] http://image-net.org/

[6] http://cs231n.github.io/convolutional-networks/

[7] https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html

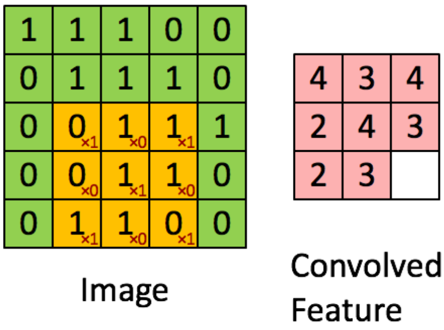

Figure.7. Convolution

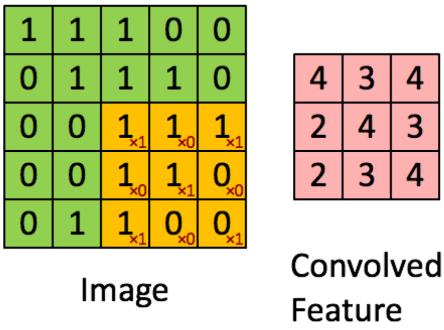

Figure.8. Convolution

Usually, deeper neural networks perform better but they are more difficult to train. ResNet use a residual learning framework to solve this problem. In a residual learning framework, when constructing a network, a shortcut connection is added showing in Figure.9. and Figure.10. The output of each layer is not the input of the traditional neural network, but the superposition of mapping and input. A residual learning framework could ease the training of networks that are substantially deeper than those used previously. ResNet explicitly reformulates the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions.[8]

In this project, the classifier is a ResNet-50 but I replace the top layer *fc1000* with one neuron.

I have tried to use simple models like SVM. I have trained a SVM classifier but it took much more time (about 2 hours) and lower accuracy (about 68%) than ResNet. SVM makes its decisions based on distance which is not good at dealing with so many images.

Figure.11. shows the structure of ResNet-50. The Resnet-50 consists of a 7x7 convolutional layer, 4 convolutional blocks, 12 identity blocks and a 1000 full connection layer.

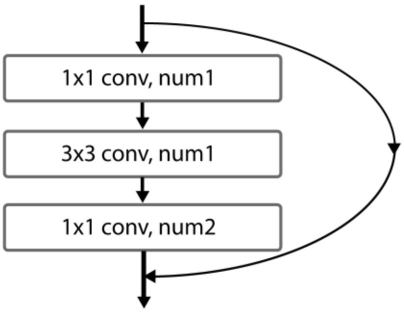

Figure.9. identity block

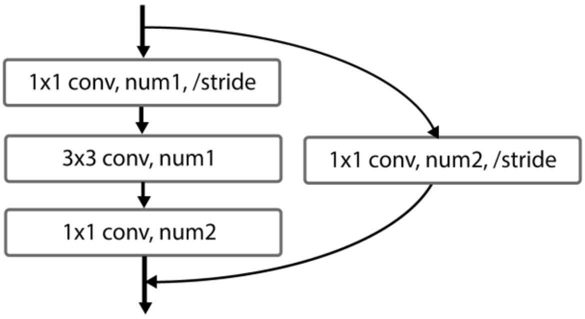

Figure.10. convolutional block

Each identity block consists of 3 convolutional layers and a shortcut without convolutional layer. The structure of identity block shows in Figure.9.

[8] https://arxiv.org/pdf/1512.03385v1.pdf

Each convolutional block consists of 3 convolutional layers and a shortcut with 1 convolutional layer. The structure of convolutional block shows in Figure.10.

Both identity blocks and convolutional blocks have 3 convolutional layers to compute the feature map. The shortcut will deliver the last block's output to the next block to avoid gradients vanishing. There is a little difference between identity blocks and convolutional blocks. Convolutional blocks have a parameter called stride and there is a convolutional layer in the shortcut which will reduce the size of feature map by half. Therefore, the initial input (224, 224, 3) becomes (7, 7, 2048) otherwise it is unpractical to conduct full connection.

ResNet-50 is good at image classification, detection and localization, so it is not a tough task for ResNet-50 to distinguish cats or dogs from each other. Normal deep convolutional neural networks also work when deal with such jobs. Usually, the deeper the convolutional neural network is, the better the accuracy shows. But when the network goes deeper again, the accuracy lowers because the gradient disappears. In ResNet-50, every identity block and convolutional

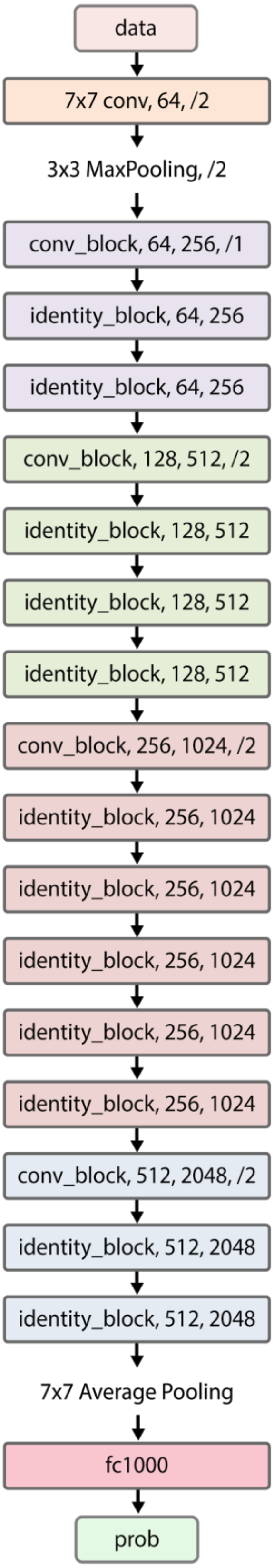

Figure.11. ResNet-50

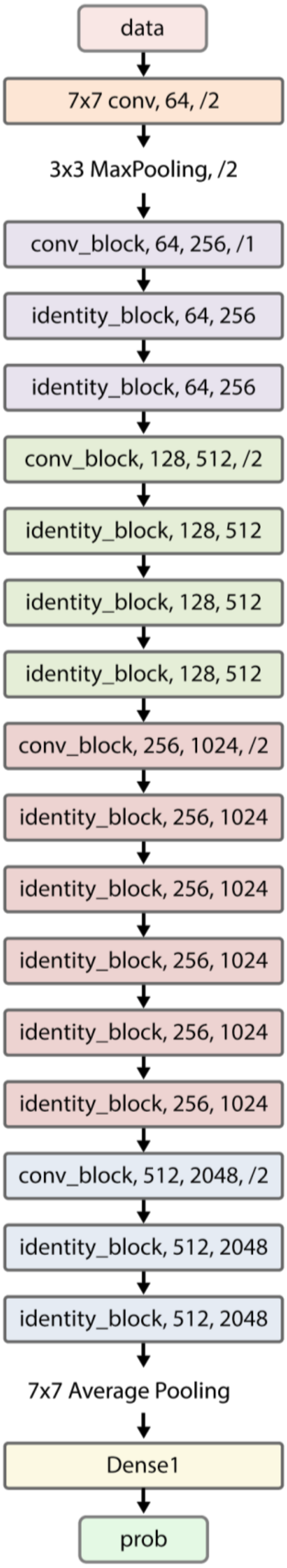

Figure.12. ResNet-50 for Cats.Vs.Dogs

block has a shortcut. Therefore, ResNet-50 with 49 convolutional layers still does a good job and better than other networks.

In this project, the network is named *ResNet-50 for Cats.Vs.Dogs*. using the structure of ResNet-50 and replacing the top layer from 1000 full connection to one neuron because the goal is to distinguish cats or dogs.

I also load the weights trained by ImageNet. This is a technique named t*ransfer learning*. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned[9]. By this way, the classifier becomes an experienced network. So, when it starts to learn images of cats and dogs, it will generalize much more typical and representative features.

Benchmark

The 1-crop validation error of ResNet-50 on ImageNet is 24.7%.[10] In other words, the accuracy is 75.3%, but the result is based on ImageNet which is much more challenging.

In this project, the goal is to keep accuracy above 90% and the prediction time less than 0.05s per image which is both effective and efficient. The log loss less than 0.69315 which is the all 0.5 benchmark on kaggle[11].

III. Methodology

Data Preprocessing

The preprocessing consists of the following steps:

1. The images in *train* folder are divided into a training set and a validation set.
2. The images both in training set and validation set are separately divided into two folders -- *cat* and *dog* according to their labels.
3. The sizes of the images are resized to 224*224.
4. The RGB color values of the images are rescaled to 0~1.

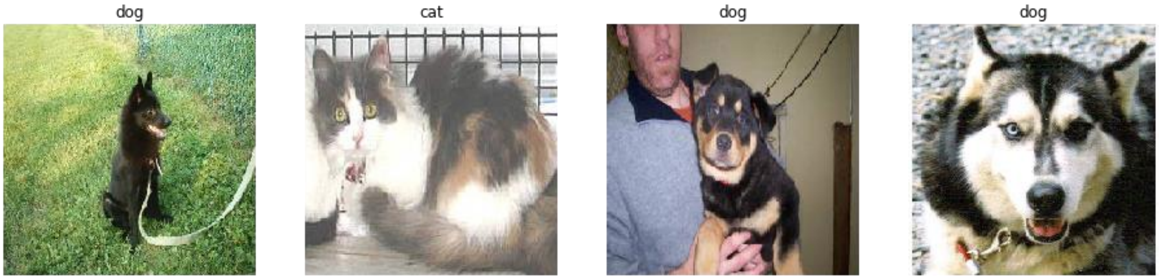

Figure.13. Images from the training set after being resized

[9] http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf

[10] https://github.com/KaimingHe/deep-residual-networks

[11] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard

Implementation

The implementation contains the following main steps:

- ❖ Build the structure of ResNet-50 for Cats.Vs.Dogs (The ResNet-50 for Cats.Vs.Dogs. consists of a 7x7 convolutional layer, 4 convolutional blocks, 12 identity blocks and a dense layer.)
    - ➢ Define identity block.
    - ➢ Define convolutional block.
    - ➢ Build the structure of ResNet-50 without top layer.
    - ➢ Load weights.
    - ➢ Add top layer to ResNet-50.
    - ➢ Setup training attribute.
    - ➢ Compile the model.
- ❖ Train ResNet-50 for Cats.Vs.Dogs.
- ❖ Save the best model.
- ❖ Use the best model to predict images.

I use keras to build the structure of ResNet-50 for Cats.Vs.Dogs. Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano[12]. Keras make it more convenient to design convolutional networks.

The weights of ResNet-50 for Cats.Vs.Dogs. without top layer were downloaded from keras and trained by ImageNet. The weights of top layer were trained by the dataset downloaded from kaggle.

The loss of the model is binary cross-entropy, the training method of the model is adadelta which dynamically adapts over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent[13].

I use dropout to prevent ResNet-50 for Cats.Vs.Dogs. from overfitting. Dropout will randomly cut up a certain percentage connection of the full connection of the top layer.

Refinement

The refinement made in this project was that I was not just build the structure of ResNet-50 and trained the classifier directly by the dataset downloaded from kaggle, instead I use transfer learning to improve the training speed and accuracy.

An initial solution to distinguish cats and dogs from each other is using a ResNet-50 with random weights. The ResNet-50 is trained by the dataset downloaded from kaggle and without transfer learning.

[12] https://keras.io/

[13] http://www.matthewzeiler.com/pubs/googleTR2012/googleTR2012.pdf

In order to obtain the val_acc (validation accuracy) above 90%, I have tried to train 200 epochs with 2048 samples per epoch. The plot in Figure.16. shows that after training 200 epochs the validation accuracy reaches to 0.89 and the training accuracy is above 0.93. It has not reached the upper limit, but the classifier is overfitting. So, I think there are better solutions which will achieve higher accuracy and cost less training time.

In order to make a refinement, I load weights trained by ImageNet, freeze the weights except the top layer, training the top layer of the network by the dataset download from kaggle. I have tried 40 epochs with 2048 samples per epoch to find the best model and the best validation accuracy reaches up to 0.98. It is almost a perfect result. The validation accuracy and loss of ResNet-50 with transfer learning shows in Figure.17. The training accuracy is less than 0.95 which means it is not overfitting.

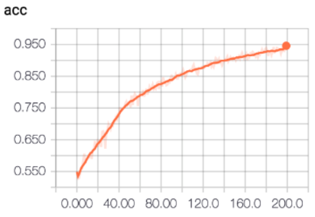

Figure.14. the training accuracy of ResNet-50 without transfer learning

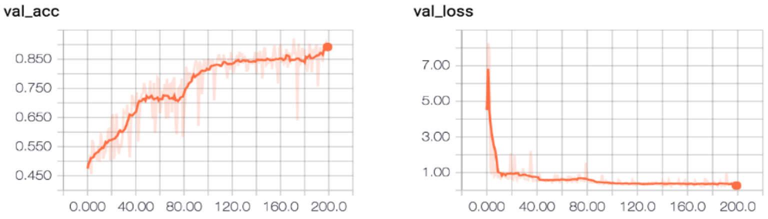

Figure.15. The validation accuracy and loss of ResNet-50 without transfer learning

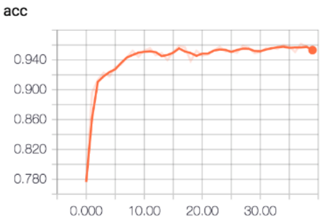

Figure.16. the training accuracy of ResNet-50 with transfer learning

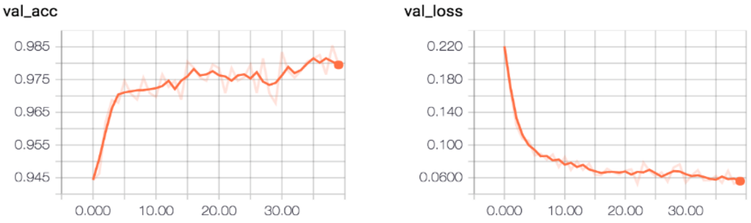

Figure.17. The validation accuracy and loss of ResNet-50 without transfer learning

IV. Results

Model Evaluation and Validation

As mentioned in the Benchmark, the goal is accuracy is above 90% and the prediction time less than 0.05s per image which is both effective and efficient.

According to the comparison in *Refinement*, ResNet-50 with transfer learning is good enough for the problem in this project. The ResNet-50 for Cats.Vs.Dogs. consists of a 7x7 convolutional layer, 4 convolutional blocks, 12 identity blocks, a dense layer and loading weights trained by ImageNet.

I did not change the basic structure of ResNet-50 like the identity block, the number of convolutional kernel or the width, height, depth of neurons, because I need to use the weights trained by ImageNet, but I replaced the top layer fc1000 with one neuron to solve the problem in this project. Batch size and training epochs (2048 samples per epoch) will influence the model. A large batch size requires pretty large memory and a small batch size might lead to overfitting, so the batch size in this project is 16. Too many training epoch will cause overfitting but if the training epoch is not enough I cannot get the best model. Therefore, I trained enough epochs (40 epochs) but I did not save the result after training 40 epochs. I saved the best model during the training process.

Figure.16 and Figure.17. shows that at the first 10 epochs, the ResNet-50 with transfer learning learned faster because the training accuracy and validation accuracy increased faster and log loss decreased rapidly. After 10 epochs, the speed of accuracy increasing and log loss decreasing tended to be gentle because the classifier has learned most of the features.

After 5 epochs' training, the validation accuracy is above 95% and the loss lowers than 0.1. After 40 epochs' training, the training accuracy shows in Figure.16. is less than 0.95 and the validation accuracy is above 0.98 shows in Figure.17. Because the curve of validation accuracy always higher than the curve of training accuracy and they did not converge, the classifier is not overfitting.

I also submitted the *result.csv* which contains the prediction of all images in test dataset on kaggle and the score is 0.08708.

In order to verify the robustness of the model, I use another dataset called *The Oxford-IIIT Pet Dataset*[14] to training the model and calculating validation accuracy. There are 2371 images of cats and 4978 images dogs. This is an unbalanced dataset which means a little challenge for the model. I use 90% images in the dataset to train the model and the rest to test. Figure.18. shows the validation accuracy is above 98% and the log loss is blow 0.06 after 38 epochs. When the dataset change, the model still performs well.

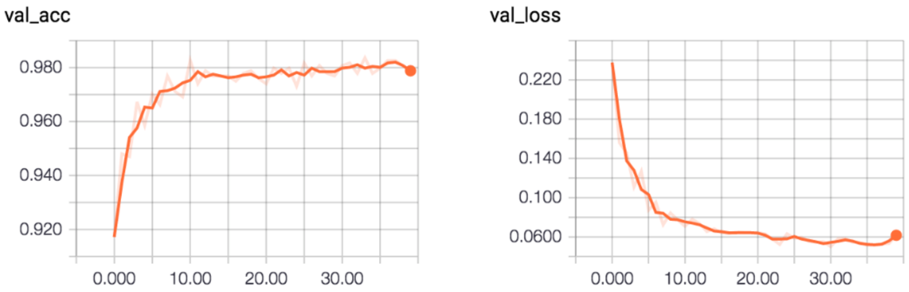

Figure.18. The validation accuracy and loss of ResNet-50 with dataset from Oxford

[14] http://www.robots.ox.ac.uk/~vgg/data/pets/

Justification

The plot in Figure.17. shows the best validation accuracy of ResNet-50 for Cats.Vs.Dogs is 98.54% which has achieved the goal.

The total prediction time of 12500 images is 68s, so it costs 0.00544s per image. The classifier ran on a computer with the following components:

1. CPU is i7 6700k
2. GPU is GTX 980 Ti
3. Memory size is 32GB.

20 random images in the *test* folder predicted by the classifier are shown in Figure.19. and each of them was predicted correctly.

V. Conclusion

Free-Form Visualization

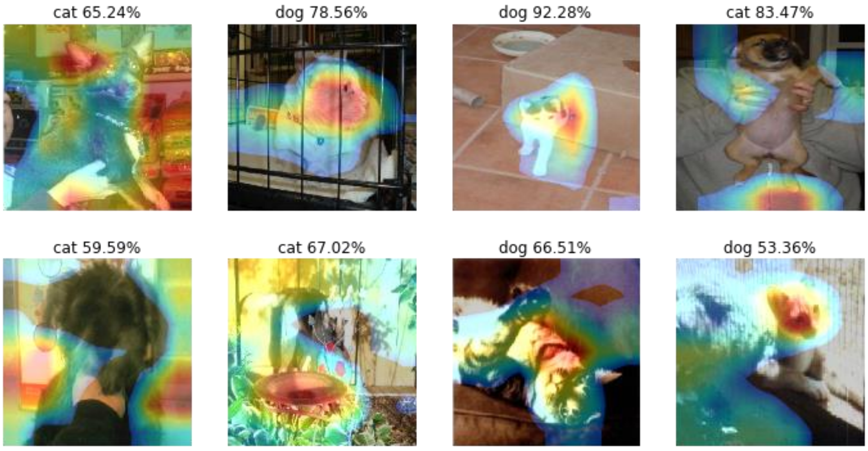

Figure.19. Random prediction result -- wrong

8 wrong prediction the classifier made shows in Fogure.19. From image (1,1) the classifier gave an answer of cat because the dog’s ear is similar to a cat ear. Other reason includes the low pixels, sufficient brightness, different postures of cats or dogs and so on.

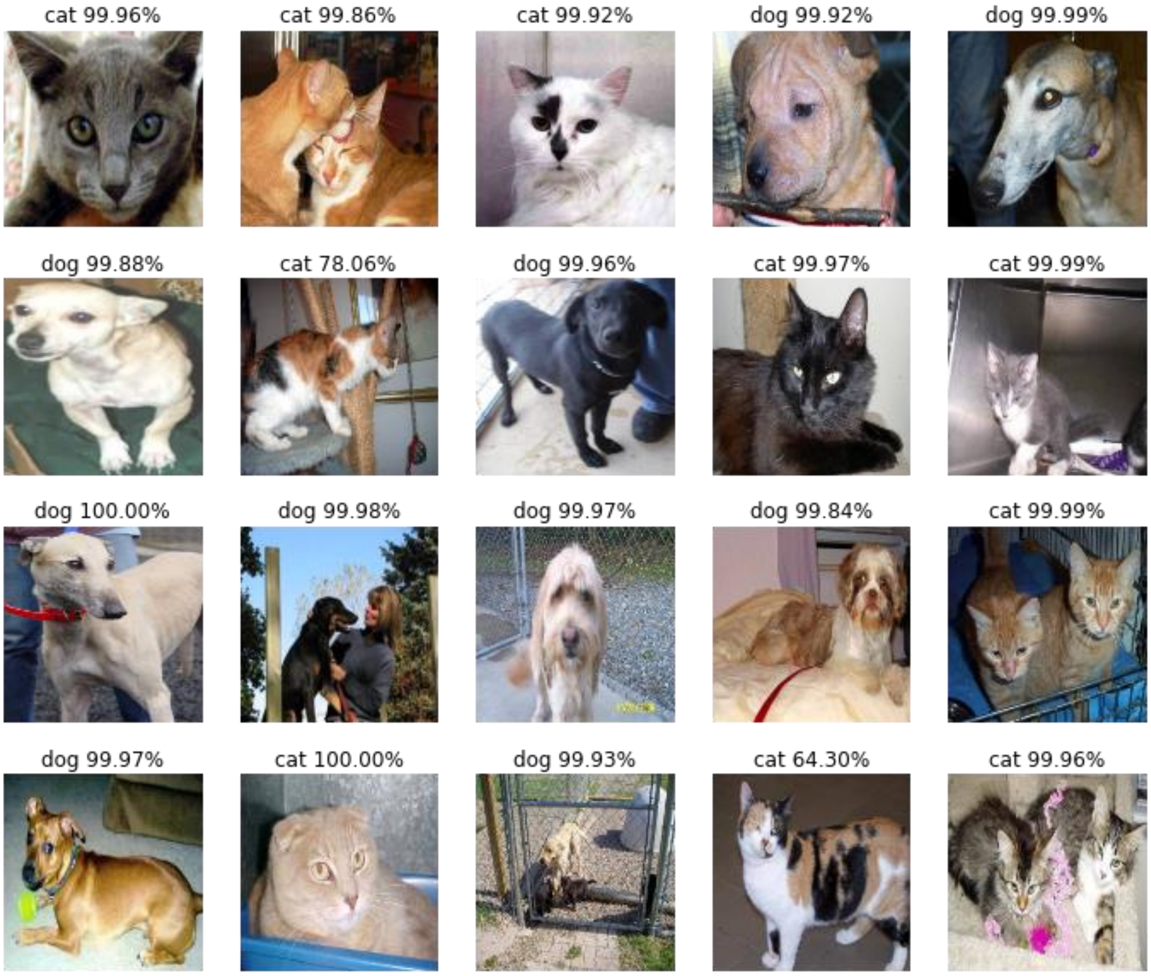

Figure.20. Random prediction result -- right

20 random images in the *test* folder predicted by the classifier are shown in Figure.20. and each of them was predicted correctly. Although the classifier could not give a 100% answer, it has done a good job at distinguishing cats or dogs from each other.

Reflection

The key steps to be done in this project is just like doing other CNN projects. The goal is to tell the it is cats or dogs in the input image.

First, find the dataset I need, download and process it. In this project, I chose the dataset from kaggle which collected a lot of daily images about cats or dogs. The preprocessing steps included dividing the images both in training set and validation set separately into two folders *cat* and *dog* according to their labels, resizing and rescaling the images.

Second, find the network which is suitable to solve this problem and build the classifier. I selected ResNet-50 and replaced the top layer from fc1000 to one neuron to solve the problem in this project. Some little techniques like transfer learning are used to making improvements.

Then, try to train the classifier with different epoch find the best model and save it.

Finally, use the best model to make predictions.

The most difficult part in this project is also the most interesting part. At first, I did not try transfer learning. I just used the dataset to train the whole ResNet-50. I tried and trained very hard but the improvement was slow, so I tried to draw the feature heatmap to understand how the classifier made its decisions.

The shape of the output of the base model is (7, 7, 2048). The shape of the weights of full connection is (2048, 1). In order to draw the heatmap, I calculated the Class Activation Mapping[15] of the output of the network using the formula below then used OpenCV to visualize the result.

$$cam = (P - 0.5) * output * w$$

- cam: class activation mapping
- P: the probability of cats or dogs
- output: the output of base model
- w: the weights of the full connection

Figure.21. shows the feature heatmap created by ResNet-50 without transfer learning and Figure.22. shows the feature heatmap created by ResNet-50 with transfer learning. It is easy to recognize from Figure.20. that the classifier recognizes a cat because the image has a cat-like face or it is a dog because it has dogs' nose, dogs' eyes, dogs' ears, etc. Figure.22., shows a different perspective. From image (1,4) in Figure.22. the classifier recognized a dog because it is outside on a meadow field rather than a dog-like face or at least dog-like body. Maybe, dogs are more likely to be outside than cats, but the classifier is not satisfied. This is also why I use transfer learning to make improvements.

[15] http://cnnlocalization.csail.mit.edu/

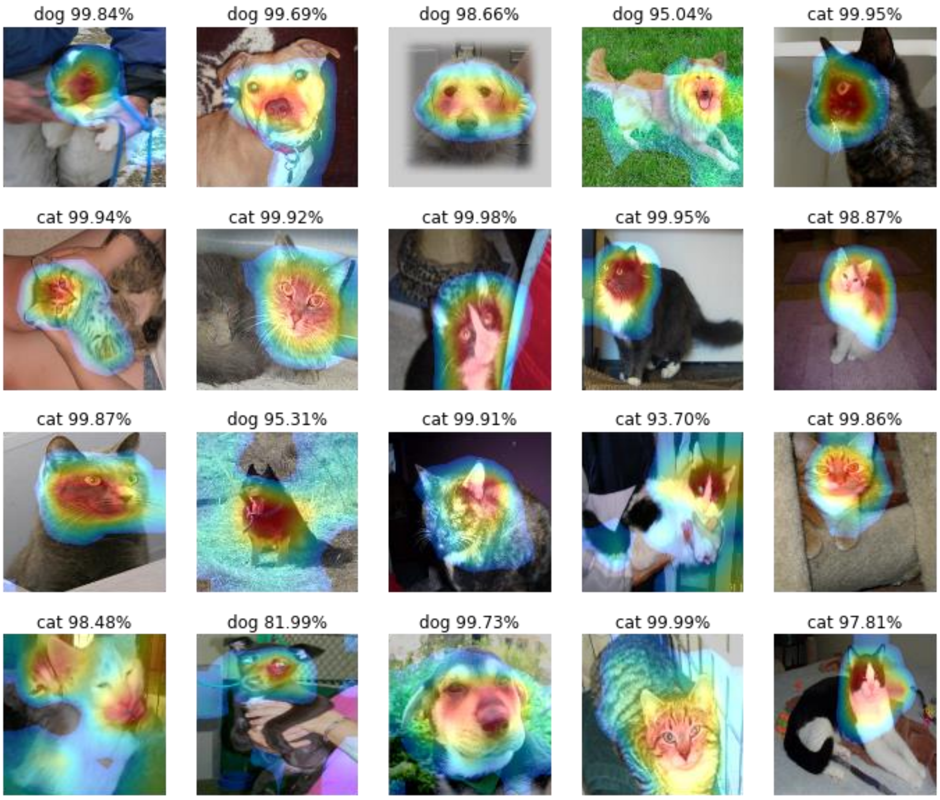

Figure.21. Feature heatmap created by ResNet-50 with transfer learning

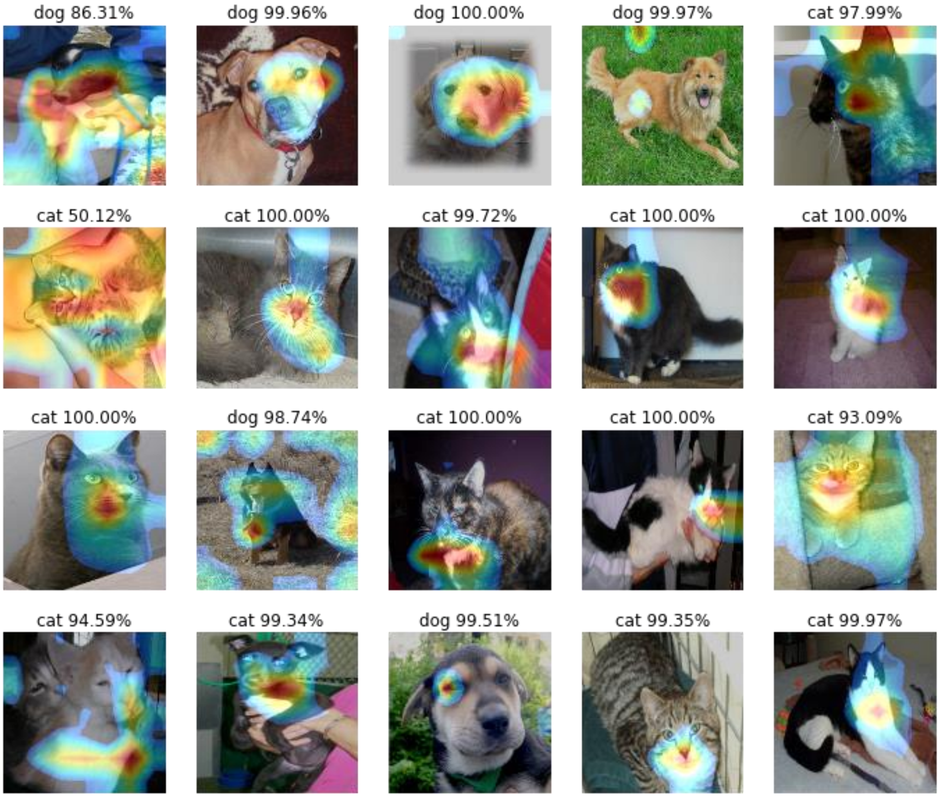

Figure.22. Feature heatmap created by ResNet-50 without transfer learning

Improvement

To achieve higher accuracy, the following improvement is worth trying.

- ❖ Use ResNet-101, ResNet-152 even 1K-layer ResNets. The 1-crop validation error on ImageNet of ResNet-101 is 23.6% while ResNet-50 is 24.7%. There is a large possibility that ResNet-101 will perform better in this project.
- ❖ More diversity of images in dataset is also helpful. Cats or dogs are taken photo of both indoors and outdoors. Images could be zoomed in or zoomed out or rotated to some angles.
- ❖ Fine-tune[16] one or more convolutional blocks.

VI. Reference

1. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
2. https://en.wikipedia.org/wiki/Supervised_learning
3. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/details/evaluation
4. https://github.com/KaimingHe/deep-residual-networks
5. http://image-net.org/
6. http://cs231n.github.io/convolutional-networks/
7. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html
8. https://arxiv.org/pdf/1512.03385v1.pdf
9. http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf
10. https://github.com/KaimingHe/deep-residual-networks
11. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard
12. https://keras.io/
13. http://www.matthewzeiler.com/pubs/googleTR2012/googleTR2012.pdf
14. http://www.robots.ox.ac.uk/~vgg/data/pets/
15. http://cnnlocalization.csail.mit.edu/
16. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[16] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html