**Machine Learning Engineer Nanodegree**

**Capstone Project**

yfchiena

January 13, 2019

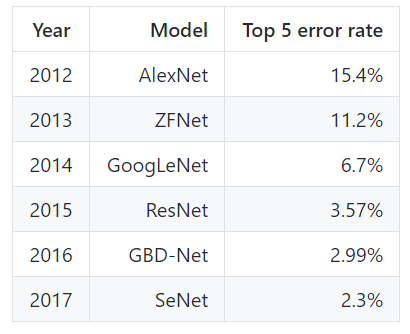
**I. Definition**

**Project Overview**

**Dogs vs. Cats** is a competition topic for Kaggle in 2013. The goal is to train a model to distinguish whether it is a cat or a dog from a given picture. This is a problem in the field of computer vision, and also a binary classification problem.

Deep learning is one of the most important breakthroughs in the field of artificial intelligence in the past decade. It has achieved great success in speech recognition, natural language processing, computer vision, image and video analysis, multimedia, and many other fields.

Hinton's research team proposed the AlexNet architecture and won the ImageNet Image Classification Competition in 2012, which the accuracy is more than 10% above the second place. Since then, the annual champions are all methods of neural networks.



**Problem Statement**

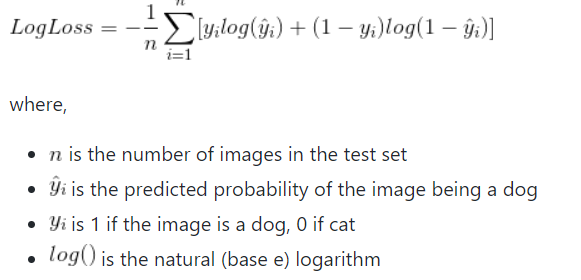
Input: A color photo.

Output: The probability (0~1) of a cat or a dog.

The project needs to identify cats and dogs, which is essentially a binary classification problem. Corresponding to supervised learning is to use the existing pictures with labels, and after the training is completed, the unlabeled pictures are classified. Therefore, it is also possible to solve this problem using a supervised learning method such as SVM. But due to the excellent performance of deep learning, I will choose the deep learning method to complete this project.

**Metrics**

A standard evaluation formula was proposed from Kaggle.



'Log loss' is also called 'Logistic regression loss' or 'Cross-entropy loss', which is one of the commonly used evaluation methods in classification problem.

**II. Analysis**

**Data Exploration and Exploratory Visualization**

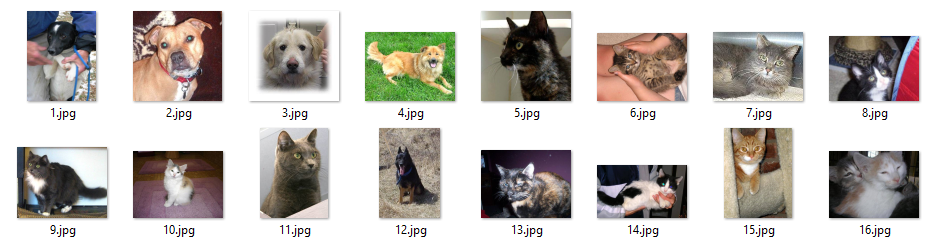
Project data sets can be downloaded from Kaggle.

The training set has a total of 25,000 images (12,500 cats and 12500 dogs), which are distinguished by file names. The tag name of each photo contains the tag of dog or cat.



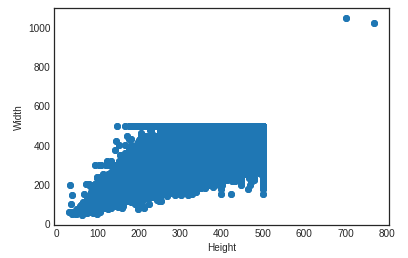
**Training set**

The test set has a total of 12,500 images and there are no tags in the file names.



**Testing set**

You can see the distribution of the image size as shown below. The distribution shows that the image size of the training set is scattered. To make the images size are consistent, I will use Keras ImageDataGenerator to solve this problem.



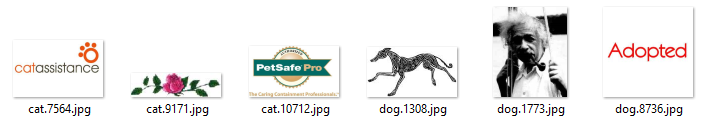
The photos were taken in everyday life. Shooting techniques are free, taking the picture of the dog as an example.

Dogs appear alone, multiple dogs, humans enter the mirror, etc.



One dog Multi-dogs Dog with human

Due to the data is presented as an image, so it can be quickly swept by the naked eye, and some abnormal images are obviously found.



**Algorithms and Techniques**

Cat and dog recognition is a binary classification problem. Along with the increase of data sets and the improvement of computing power, especially the explosive improvement of GPU computing power, the deep learning using convolutional neural network solves the problem of picture recognition. This project will use tensorflow and keras to build a learning model that is quick and easy to use.

CNN (Convolutional Neural Network) is one of the most powerful deep learning neural networks, especially in image recognition applications. CNN's design prototype originated from LeNet5. It was mainly used for text recognition. Later, people continued to optimize the structure and performance of the network to form the CNN that we use today.

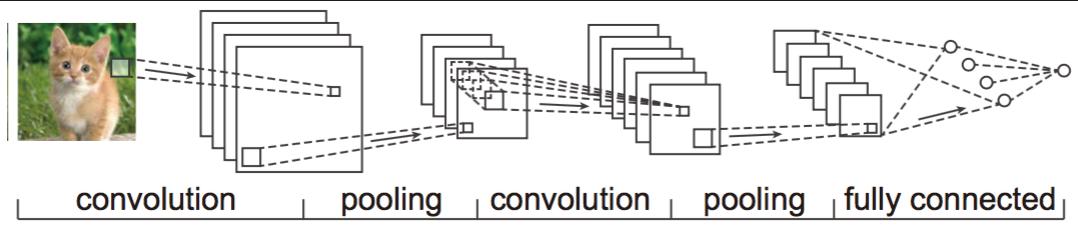
CNN's advantages in image recognition are mainly reflected in:

\* Reduce the number of weights in the calculation process

\* It can resist slight distortion, deformation and other disturbances in object recognition

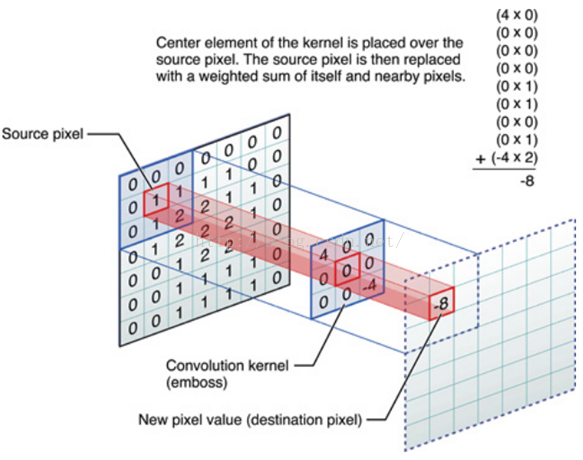
\* The recognition of an object is not affected by the positional change of the object in the picture

The main components of CNN are convolution layer, pooling layer, fully connected layer. By continuously reducing the dimensions of the data, it can eventually be used to train. Below I will introduce the knowledge of 'Convolution layer', 'Pooling layer', 'Dropout layer' and 'Transfer learning'.



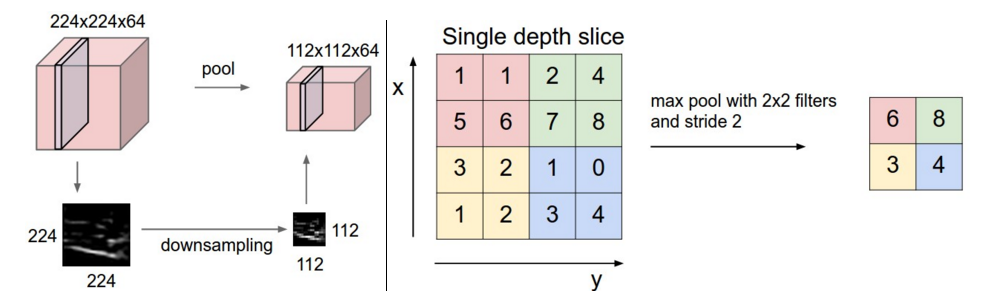
**(1) Convolution layer**

Through the convolution operation, the computationally intensive image recognition problem is continuously reduced, and finally it can be trained by the neural network. Intuitively, the convolutional layer can be thought of as a series of training/learning filters. A convolution filter slides over each pixel to form a new output data (feature map).



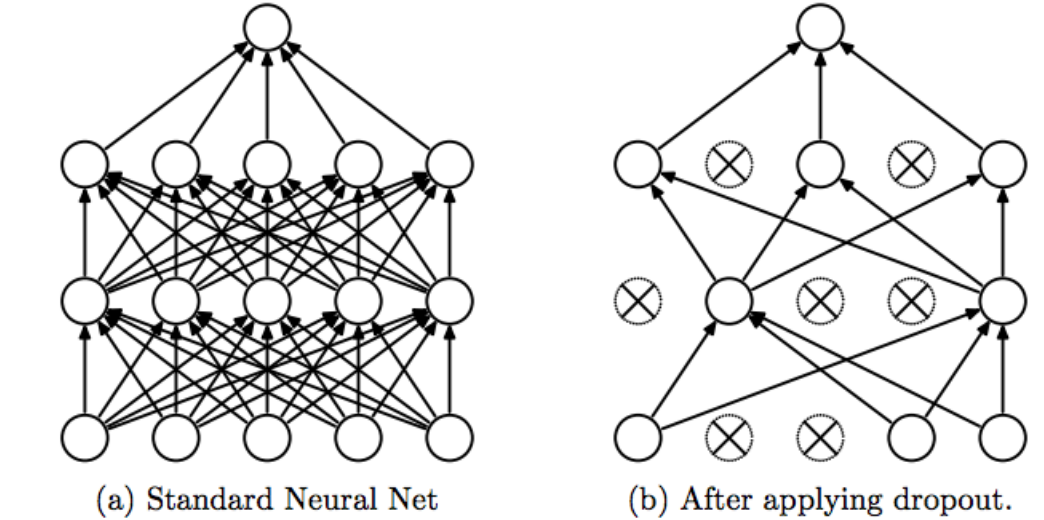
**(2) Pooling layer**

Theoretically, the data after the convolution process can directly use for training, but the amount of calculation is still too large. In order to reduce the amount of calculation and improve the generalization ability of the model, we will pool it.



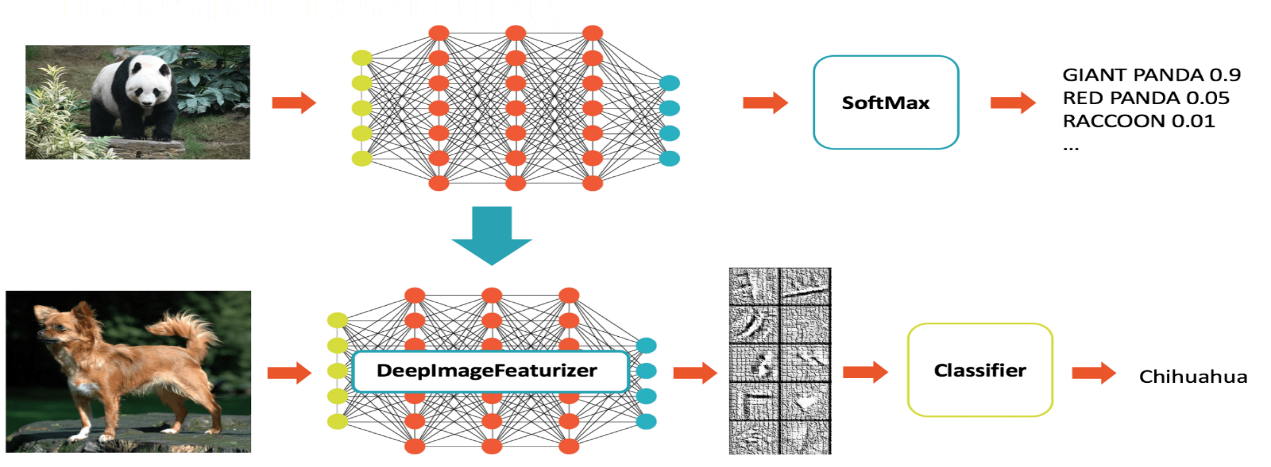
**(3) Dropout layer**

During the training of the deep learning network, the neural network unit is temporarily discarded from the network with a certain probability to prevent over-fitting problems. This way can reduce the interaction between feature detectors and enhance the generalization ability of the model.



**(4) Transfer learning**

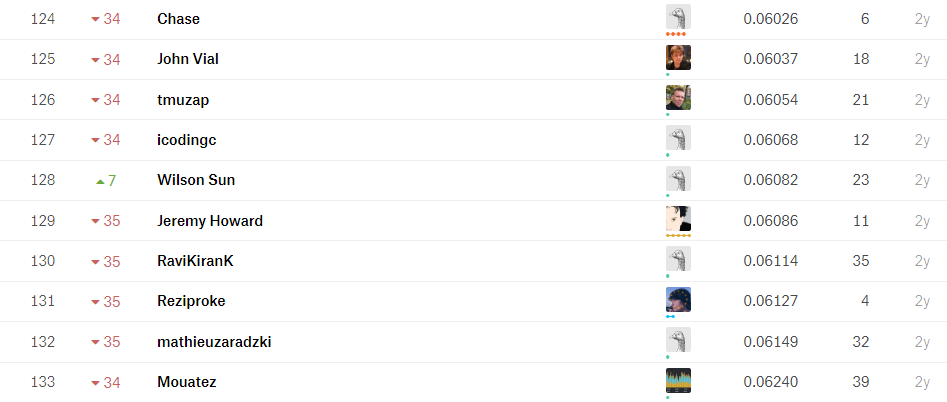
As the name suggests, it is to migrate the learned model parameters to the new model to help the new model training. Considering that most of the data or tasks are related, we can use the migration learning to share the model parameters (also known as the knowledge learned by the model) in a certain way to the new model to speed up and optimize. The learning efficiency of the model does not have to learn from zero like most networks. For example, if you have already played Chinese chess, you can learn chess in analogy. If you have learned English, you can learn French in analogy. Everything in the world has commonalities. How to reasonably find similarities between them and use this bridge to help learn new knowledge is the core issue of migration learning.



**Benchmark**

In this project, I decided to adopt the classic ResNet model. Since its introduction in 2015, ResNet has won the first place in the ImageNet competition classification task because it is “simple and practical”, and many methods are based on ResNet50 or ResNet101. On the basis of the completion, detection, segmentation, identification and other fields have a wide range of applications.

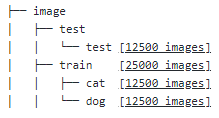
On Kaggle, there are total of 1,314 teams participated in the competition. In the Chinese version of the capstone project, there is a threshold for students to pass. The minimum requirement is reaching the top 10% (1314\*0.1 = 131) of the Kaggle Public Leaderboard. That is the Logloss on the Public Leaderboard is below 0.06127.



**III. Methodology**

**Data Preprocessing**

\* Split the training set into dog/ and cat/

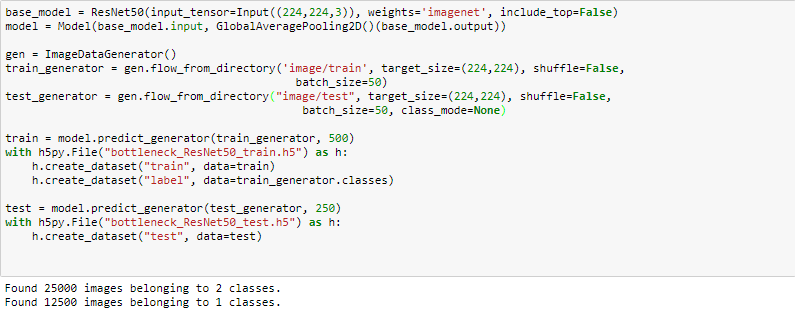


\* Use ImageDataGenerator to unify the image size into (224,224) and cat label as 0, dog label as 1

**Implementation**

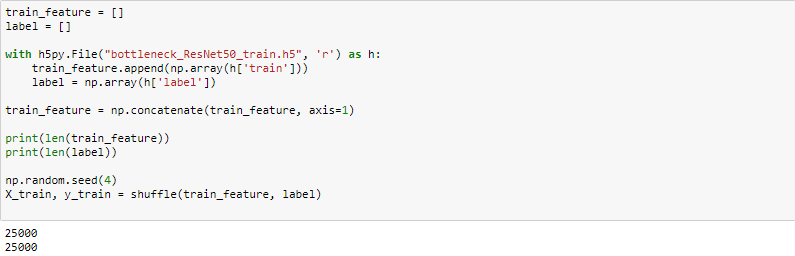
1. Export bottleneck feature

Use Keras' pre-training model ResNet50 to extract features and save for subsequent training and testing. Use GlobalAveragePooling2D to directly average each activation map of the convolutional layer output, otherwise the output file will be very large and easy to overfitting. Then we use the model.predict\_generator function to derive the bottleneck feature.



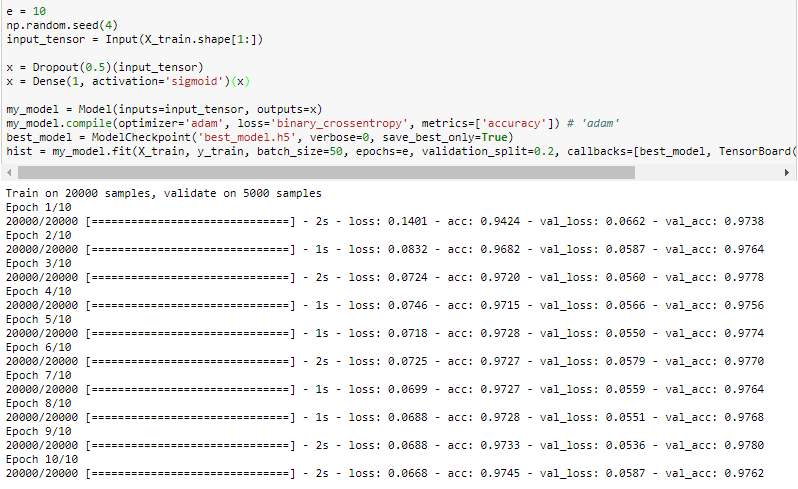
2. Load bottleneck feature

In addition, it needs to shuffle the data, otherwise we will have problems after setting validation\_split. This is because there is a trap here, the program executes the validation\_split and then shuffle, so this happens: If your training set is ordered, for example, the positive sample is in the front negative sample, and then set with validation\_split, then your validation set will most likely be a negative sample. Similarly, this thing will not be reported any errors, because Keras can't know if your data has been shuffled. If you are not shuffled, it is best to shuffle it manually.



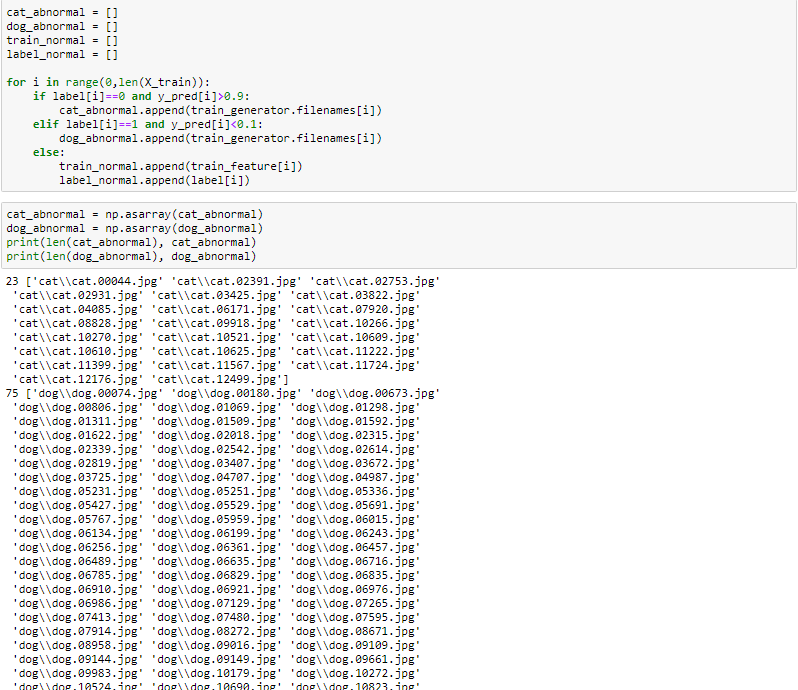
3. Construct and train the model

A part of the training set is taken out as a validation set in a certain proportion. Here we set to training set 8: Validation set 2 split ratio (0.2)

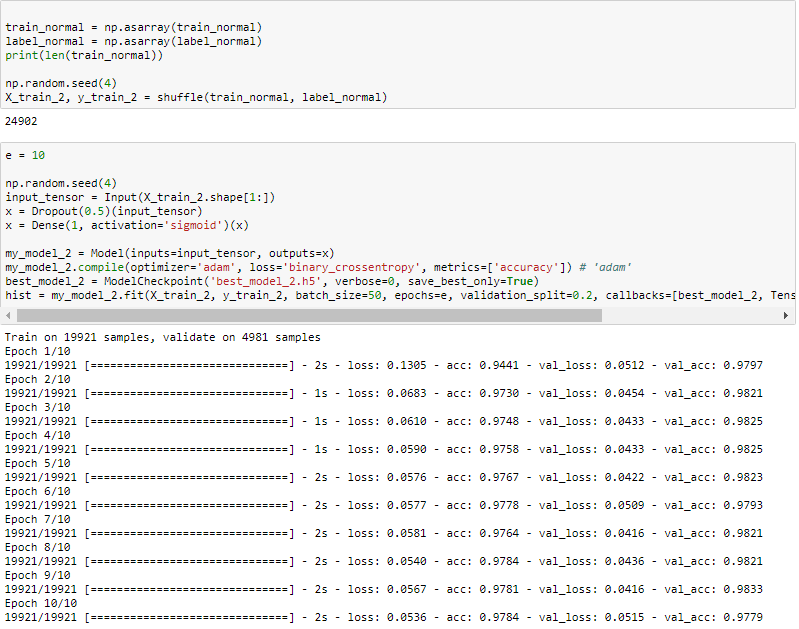


4. Remove abnormal images

Use step3 well-trained model to predict the training set and try to remove pictures with a large difference in the probability of owning the label (e.g. image is cat, but the result probability is 0.92, which classify as a dog). But even with such a strict setting, there are still many pictures that have been deleted by mistake.



5. Re-train the model

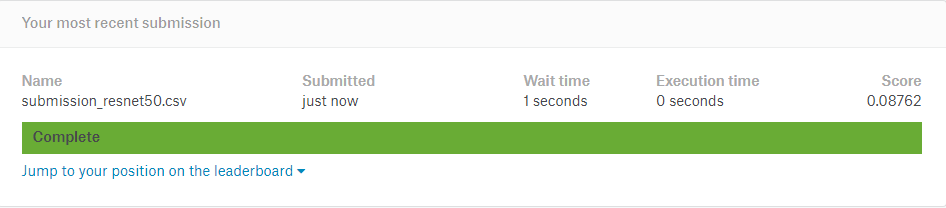


6. Predict for testing set and upload csv to Kaggle

Here we have a clip operation on the score of the result, and each prediction is limited to [0.005, 0.995] intervals. The official evaluation criterion of Kaggle is Logloss. For the prediction of the correct sample, the difference between 0.995 and 1 is very small, but for the sample that predicted the error, the gap between 0 and 0.005 is very large.



The score of the submission result is 0.08762, bigger than 0.06127.

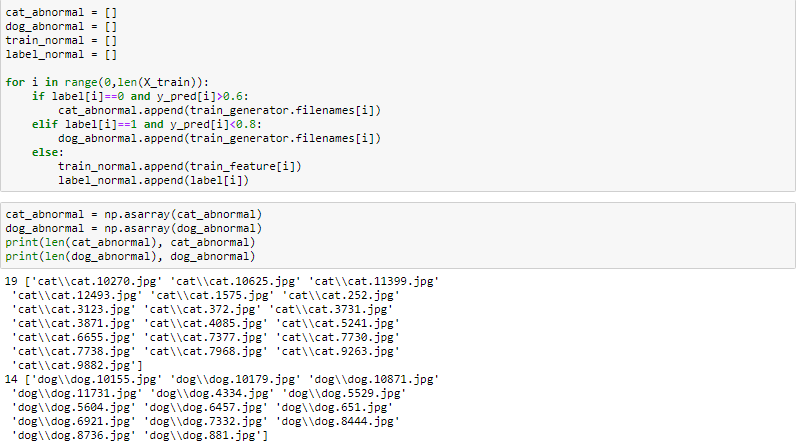


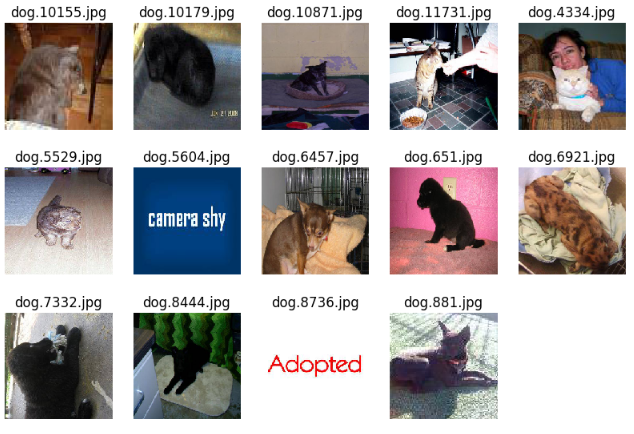
**Refinement**

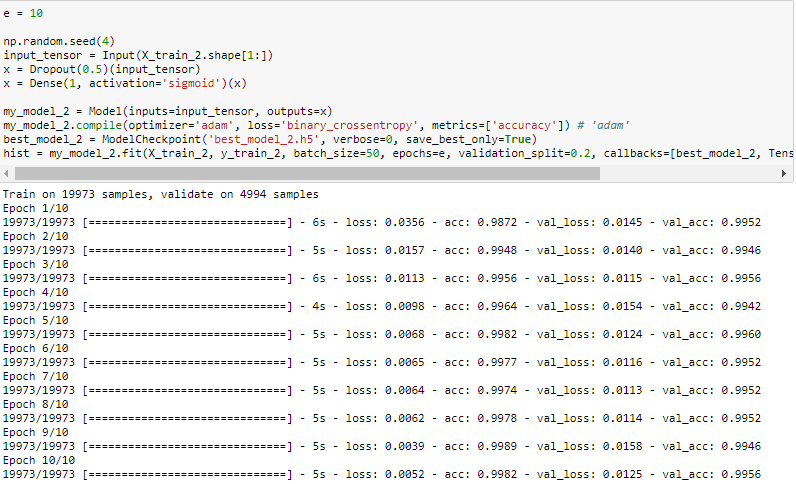
After several times changes for parameter (such as optimizer, dropout ratio etc.), the score still can't reach top 10%. Then, I found a method to "merge model" when I was looking for another solution, and the result looks very powerful. [1]

The concept is to integrate the bottleneck features of different models (the author choose ResNet50, Xception, Inception V3) into a new bottleneck.

Using the merge model, I found that even if I relaxed image filter conditions, the situation in which the picture was misjudged improved a lot.



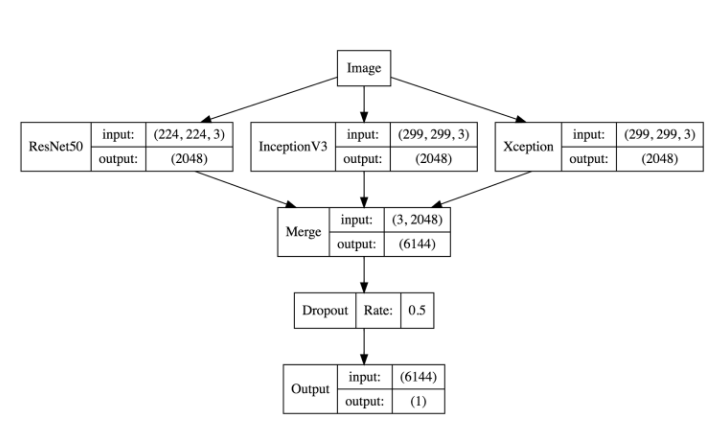




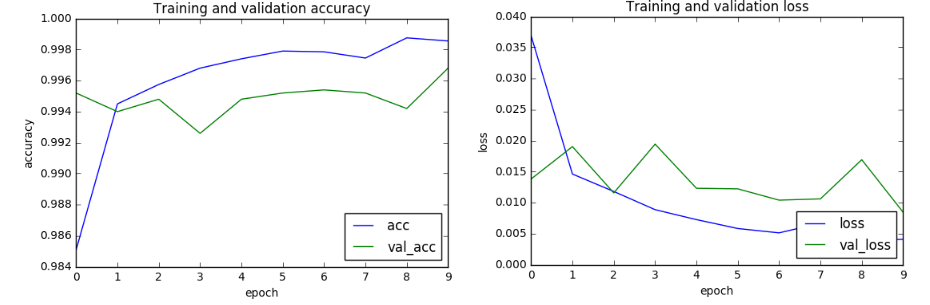
**IV. Results**

**Model Evaluation and Validation**

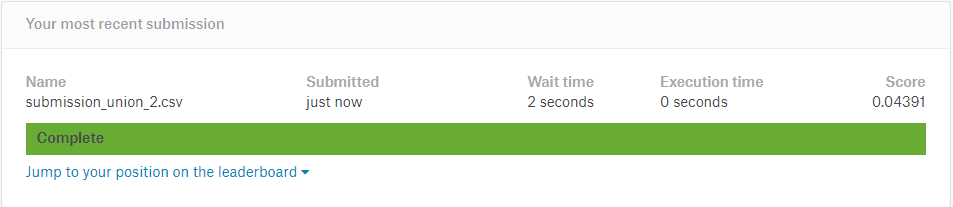
The final model is shown as below.

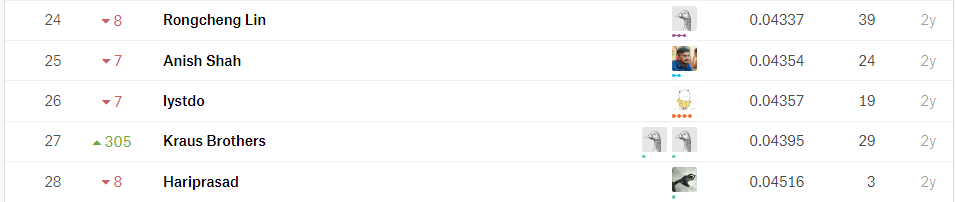


And the training accuracy increased to 0.998, training loss reduced to 0.03, validation accuracy increased to 0.995 and validation loss reduced to 0.11.



**Justification**

Finally, the score improves to 0.04391(27/1314). 



**V. Conclusion**

**Free-Form Visualization**

The above figure shows the prediction results of some test samples. It can be seen that the final model's predictions for these samples are basically correct. It can be seen from these results that the model is very capable of recognizing cats and dogs, even if many other objects appear in the picture, such as human body, cage etc.



**Reflection**

Using neural networks trained by ImageNet at the beginning of this experiment is the right choice, but the neural network of the Resnet50 may be not strong enough to meet the requirements of the project. But because of this, I learned a new method [1]. Once again prove the truth of **Unity is strength**. However, through this experiment, I also learned that it is very wise to use migration learning to solve such similar image classification problems. Keras has greatly improved the possibility of development and verification exploration.

**Improvement**

Although the initial goal has been reached, on the basis of the existing ones, there are still several ways to improve and improve the accuracy of the model:

(1) Combination of other model merge

There are still have lots of powerful model, such as VGG16, VGG19, InceptionResNet V2, DenseNet etc.

(2) Increase the train set data

We can increase the number of train samples by flipping the image, changing the brightness of the image, and so on.

(3) Better data cleaning method

(4) Fine-tune the model

**Reference**

[1] https://zhuanlan.zhihu.com/p/25978105