Image classifier—Dogs vs Cats

Name: Yukang Xu

Ⅰ. Introduction

Web services are often protected with a challenge that's supposed to be easy for people to solve, but difficult for computers. Such a challenge is often called a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIP (Human Interactive Proof). HIPs are used for many purposes, such as to reduce email and blog spam and prevent brute-force attacks on web site passwords.

Asirra (Animal Species Image Recognition for Restricting Access) is a HIP that works by asking users to identify photographs of cats and dogs. This task is difficult for computers, but studies have shown that people can accomplish it quickly and accurately. Many even think it's fun! Here is an example of the Asirra interface:

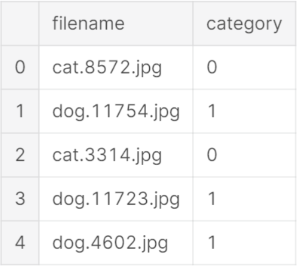
Asirra is unique because of its partnership with Petfinder.com, the world's largest site devoted to finding homes for homeless pets. They've provided Microsoft Research with over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States. Kaggle is fortunate to offer a subset of this data for fun and research.

In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.

Ⅱ. Data Analysis

1. Import and Clean data

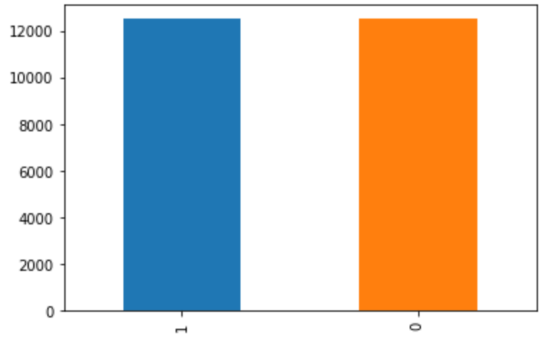
We have two folders which are dogs’ and cats’ pictures. I am trying to put them in one data frame, one column is picture name while the other column is category (cat is 0, dog is 1). Since it is easy to classify all pictures, we skip data cleaning step.



	filename	category
0	cat.8572.jpg	0
1	dog.11754.jpg	1
2	cat.3314.jpg	0
3	dog.11723.jpg	1
4	dog.4602.jpg	1

2. Explore our dataset

Let us take a look at our dataset. It seems that numbers of cats’ or dogs’ pictures are equal. Since we promise percentage of dataset is fair, it is good for our training.



3. Model Building

We will implement convolutional neural network to classify images. For convolutional neural network (detecting image), there are three different steps to get outcome after importing images.

a. convolutional layer (the convolution is performed on the input data with the use of a filter then produce a feature map; We execute a convolution by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map. In this case, we use these filters to find specific pattern like nose or ears)

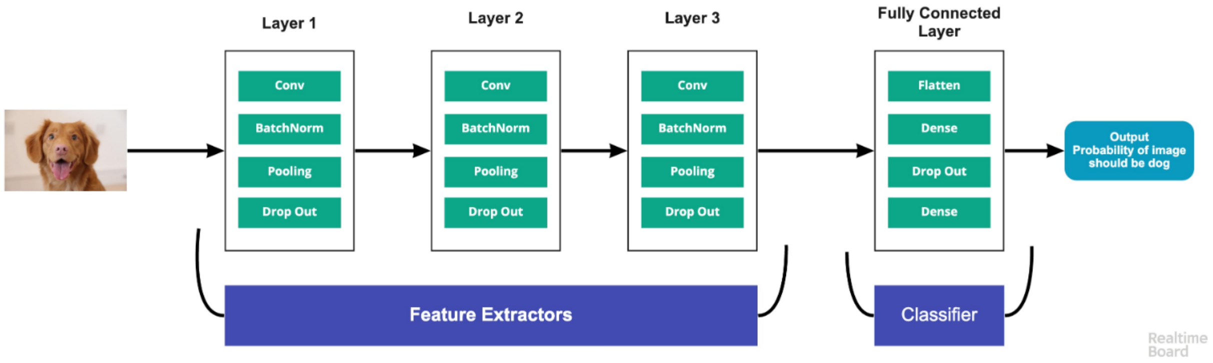
b. pooling layer (The function of pooling is to continuously reduce the dimensionality to reduce the number of parameters and computation in the network. This shortens the training time and controls overfitting. The most frequent type of pooling is max pooling, which takes the maximum value in each window. These window sizes need to be specified beforehand. This decreases the feature map size while at the same time keeping the significant information. In this case, we want to keep most important features of dogs or cats)

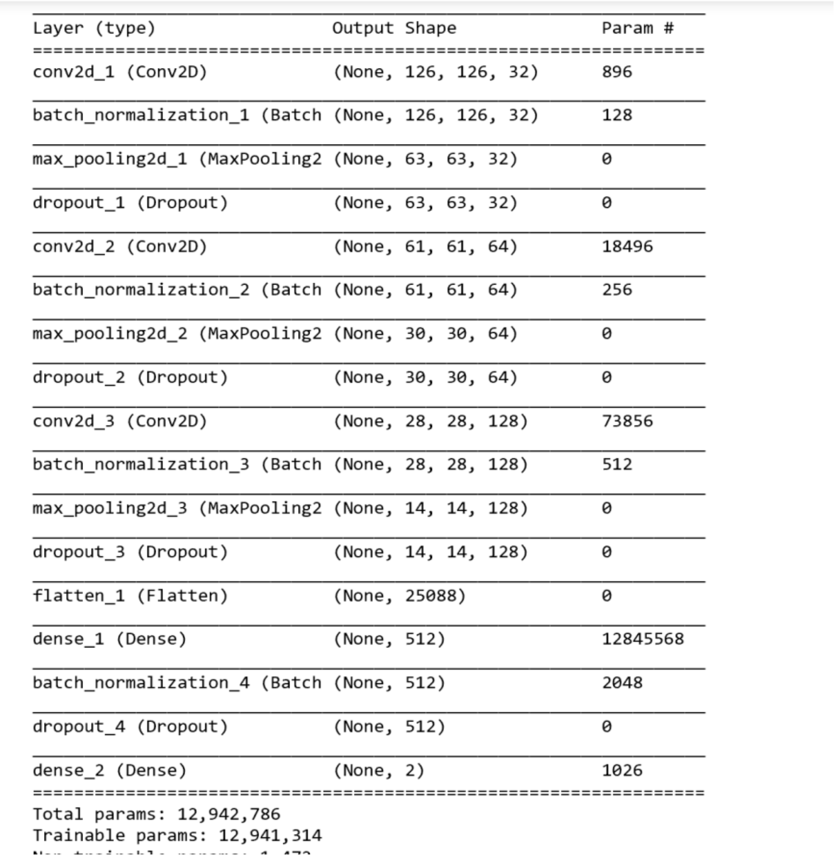
3. full connected layer (After the convolution and pooling layers, our classification part consists of a few fully connected layers. However, these fully connected layers can only accept 1 Dimensional data. To convert our 3D data to 1D, we use the function flatten in Python. This essentially arranges our 3D volume into a 1D vector. Then takes the inputs from the feature analysis and applies weights to predict the correct label)

Simple example: There are apps that allow you to take a picture of a business card. By detecting individual digits. Let's tackle a simpler version of that: Recognizing a digit: 0 or 1. OK now we have a task. If you had to explain to someone what 1 and 0 look like in terms of vertical and horizontal lines, what would you say? 1 looks like - a single vertical line. And 0 looks like - two vertical lines and two horizontal lines forming a tall rectangle. So you just need to detect specific shape in a image. The given image is nothing but a grid of pixel values. So we transfer a image into a array and then detect specific numbers. Think in terms of edges and shapes. How do we differentiate them? A dog has a longer nose, i.e. the outline of the face is typically longer. Some have longer ear. The nose is black (i.e. a concentration of pixel values is close to 0). Longer legs. And a dozen others which a CNN can internally learn and compute via filters, given thousands of labeled images of dogs and cats.

In summary, CNNs are especially useful for image classification and recognition. They have two main parts: a feature extraction part and a classification part.The main special technique in CNNs is convolution, where a filter slides over the input and merges the input value + the filter value on the feature map. In the end, our goal is to feed new images to our CNN so it can give a probability for the object it thinks it sees or describe an image with text.

In this project, we can summarize this process into a picture as follows. Then we can move on to train our model below.





```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 61, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               12845568
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 12,942,786
Trainable params: 12,941,314
Non-trainable params: 1,472
```

4. Generator Building (Data Augmentation)

One of the best ways to improve the performance of a Deep Learning model is to add more data to the training set. Aside from gathering more instances from the wild that are representative of the distinction task, we want to develop a set of methods that enhance the data we already have. There are many ways to augment existing datasets and produce more robust models. In the image domain, these are done to utilize the full power of the convolutional neural network, which is able to capture translational invariance. This translational invariance is what makes image recognition such a difficult task in the first place. You want the dataset to be representative of the many different positions, angles, lightings, and miscellaneous distortions that are of interest to the vision task.

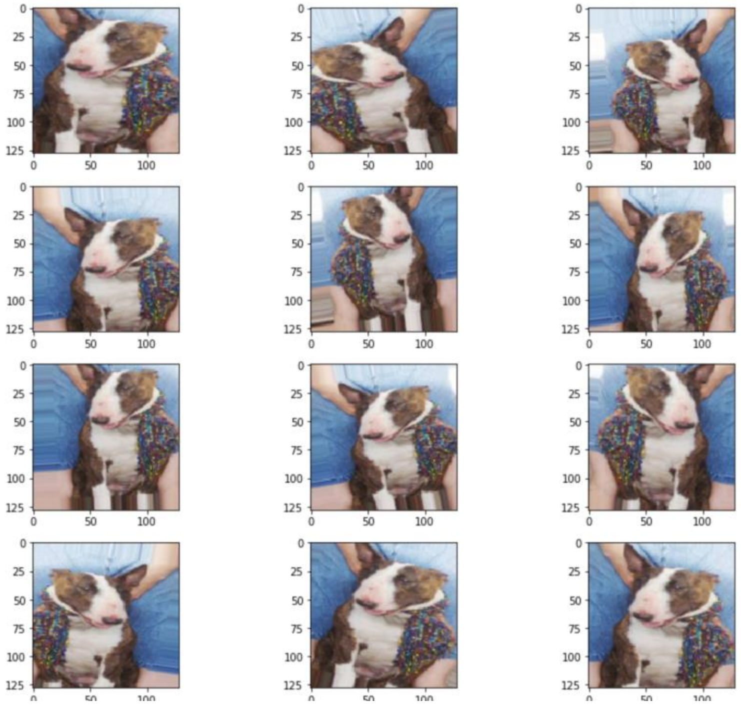
For example, if every image in your training set is perfectly centered within the frame, traditional feed forward models would be confused when the image is slightly shifted to the right relative to the background. If every picture of a cat is taken in portrait made, the model will not recognize the cat when it’s face is turned to the right.

To improve the model’s ability to generalize and correctly label images with some sort of distortion, we apply several translations to our dataset. This can be done in many ways, in this article we will focus primarily on horizontal and vertical flips, translations, color distortions, and adding random noise.

The relevance of horizontal and vertical invariance is very intuitive. Additionally, we might want to add translational shifts to images in our dataset.

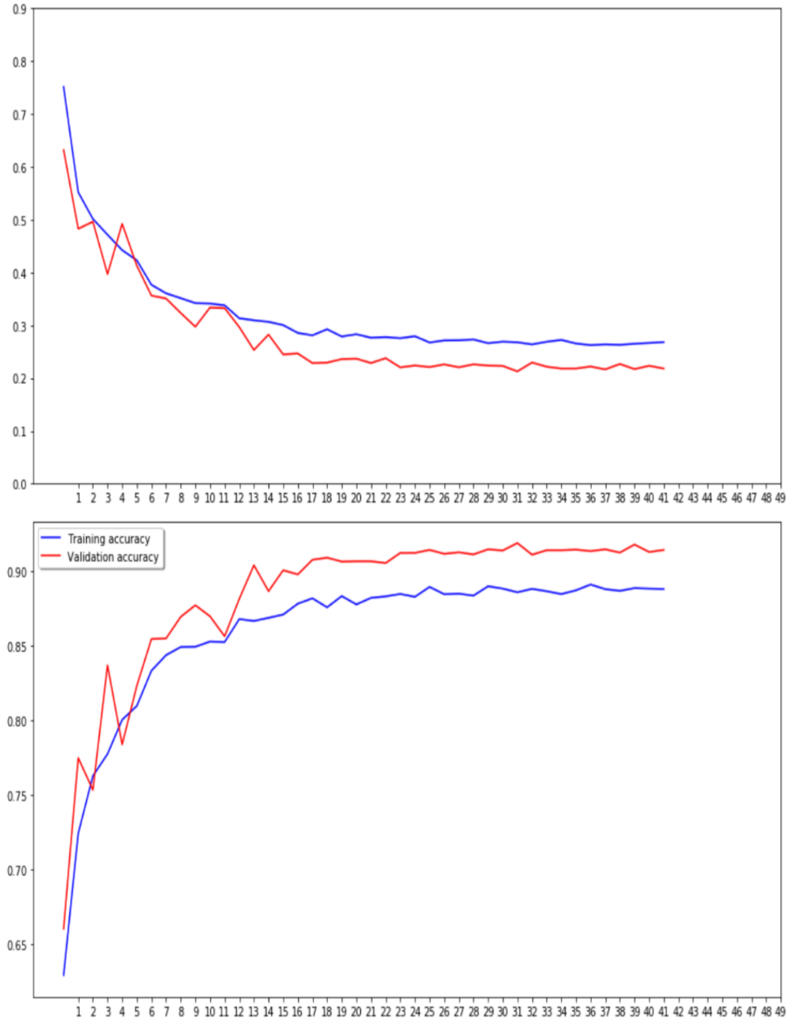
All in all, we are trying to improve the accuracy of prediction. So we randomly create more samples then increase the generalizability of the model. The way we apply this is to rotate, flip or change scales or so on.

We select one of picture to see how our generator works.



5. Model Evaluation

If we compare accuracy between training data and validation data, we can find that with more training data, accuracy of training data is similar to validation data. So our model passed our test.



Ⅲ. Prediction

For categorial classification the prediction will come with probability of each category. So we will pick the category that have the highest probability with numpy average max. We will convert the predict category back into our generator classes by using `train_generator.class_indices`. It is the classes that image generator map while converting data into computer vision.

From our preparation data part. We map data with `{1: 'dog', 0: 'cat'}`. Now we will map the result back to dog is 1 and cat is 0. Finally, let us take look at our prediction result.