Animal Image Classifier Project

Kihong Kim (Data61, Pullenvale)

Supervisor : Kusy, Brano (Data61, Pullenvale)

1. INTRODUCTION

Monitoring wild life is getting important and many people wants to collect the images and classify them from the wild. Understanding species of wild animals and behaviors of them is essential for evaluating biodiversity and changes in wild.

And also, Image Classification is an emerging technology in the machine learning and deep learning field. So, I started the Animal Image Classifier Project. The project was to setup a camera system first on-site at QCAT, later lone pine koala sanctuary, to collect and annotate images of animals passing in front of the camera. The main goal was to develop an image classifier that can detect kangaroos/deer/foxes in the field of view of the camera.

There are three major steps for developing an image classifier that can detect animals in the field of view of the camera. First, Collecting and labeling the images taken from the camera. Second, Train the images with the deep learning algorithm. In this project, I used CNN(Convolutional Neural Network) for image classification. Among the various models(methods) for image classification(i.e. VGGnet, AlexNET, Inception, ResNet). I began with Inception-V3 from Google. I could train this model with the images I collected. Third, classifying the images with the trained model. After training model, I could classify and label the images that I wanted to know what it is. But, due to the camera failure, I jumped over the step 1 and instead, I took advantage of Caltech256 dataset.

2. SYSTEM

2-1. Caltech256 Dataset

The goal of my work is to simplify and implement the process of animal identification using images from motion-triggered cameras installed in wild. The original idea was to collect images from the camera installed in wild. As equipped trap camera was needed to be fixed, I implemented this project with the dataset

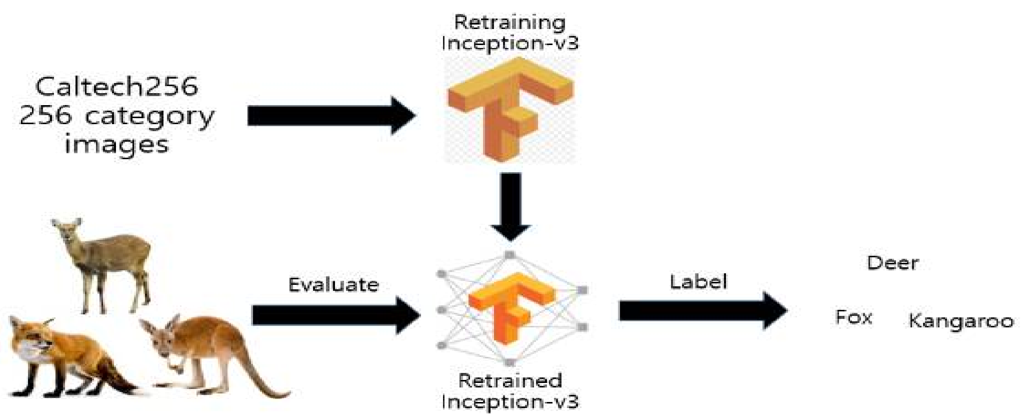
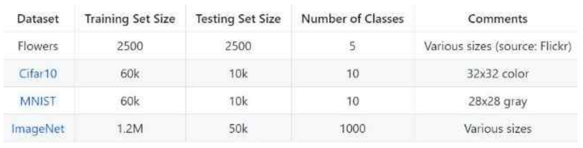


Figure 1: Training Workflow for Generating Model & Evaluating Workflow for Labeling Images

named Caltech256. Caltech256 dataset is a challenging set of 256 object categories containing a total of 30906 images, downloading examples from Google Images and then manually screening out all images that did not fit the category. I used Caltech256 dataset for training the model(Inception-V3)

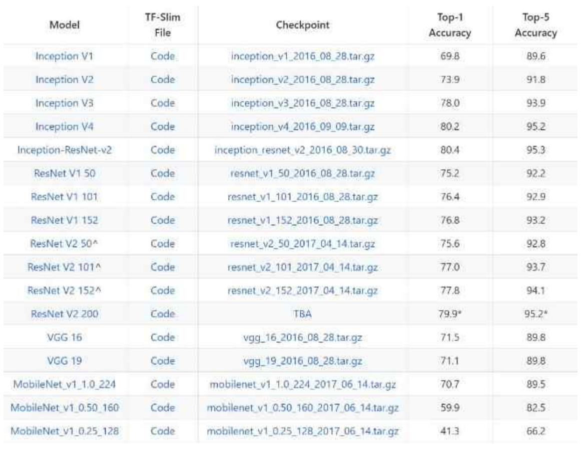
2-2. Tensorflow-SLIM

For this project, I used Python, Tensorflow and TF-Slim(Tensorflow-SLIM). Python is a language for Tensorflow and Tensorflow is a powerful library for Machine Learning and Neural Network. Also, TF-slim is a library based on Tensorflow. With TF-Slim, I could easily work with the state-of-the-art models and famous datasets as well. You can also check the datasets and CNN-models offered by TF-Slim with Figure 2 and Figure 3



Dataset	Training Set Size	Testing Set Size	Number of Classes	Comments
Flowers	2500	2500	5	Various sizes (source: Flickr)
Cifar10	60k	10k	10	32x32 color
MNIST	60k	10k	10	28x28 gray
ImageNet	1.2M	50k	1000	Various sizes

Figure 2: Datasets offered by TF-Slim



Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101^	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152^	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8
MobileNet_v1_1.0_224	Code	mobilenet_v1_1.0_224_2017_06_14.tar.gz	70.7	89.5
MobileNet_v1_0.50_160	Code	mobilenet_v1_0.50_160_2017_06_14.tar.gz	59.9	82.5
MobileNet_v1_0.25_128	Code	mobilenet_v1_0.25_128_2017_06_14.tar.gz	41.3	66.2

Figure 3: CNN Models offered by TF-Slim

2-3. Implementation

Workflow of my project is depicted in Figure 1. As I explained in 2-1 and 2-2, I developed the system with TF-slim and Caltech256. From now on, I will deal with the steps I used TF-Slim. I have used TF-Slim in 3 major steps.

First, Transforming images into TFRecord File. For each dataset, I needed to convert the raw data to TFRecord format which is available on the Tensorflow.

Second, Retraining the model. There is a two ways that I can train models. One is training a model from scratch. This means that you train the model with your own data from the beginning. This process may take several days, depending on your hardware setup. The other is fine-tuning a model from an existing checkpoint. Rather than training from scratch, I can start from a pre-trained model and fine-tune it. This means that we can use the calculated weights and parameters of the model by using existing checkpoint. By doing fine-tuning, we can easily reduce the time for training and get the reasonable result. When fine-tuning a model, we need to be careful about restoring checkpoint weights. In particular, when we fine-tune a model on a new task with a different number of output labels, we won't be able to restore the final logits(classifier) later. So keep in mind that warm-starting from a checkpoint affects the model's weights only during the initialization of the model. So after starting training, we should only train the new layers. Among the two methods, I chose fine-tuning the model. Fine-tuning is the faster than training from the scratch. And I could also get even better result as well.

Third, Labeling images. After I trained the model, I could label the images by using the weights of trained-model. First of all, I labeled the Image one by one. After that, I added some code so that I can automatically classify all of the images in one folder.

I followed these 3 steps to train model and make model classify the images that I want.

3. EVALUATION

Accuracy	Top5	Top1
Fine-tune	93.5%	80.6%
Training from scratch	82.1%	65.0%

Table 1: Top1 & Top5 Accuracy of two methods

Table 1 is the result of two methods(Fine-tuning and Training from scratch). Comparing those two methods, I chose to fine-tune the model due to the faster training and higher accuracy.

Accuracy	Cor	False
Kangaroo	19	1
Duck	18	2
Dog	17	3

Table 2: Correct & False of 20 images, 3 categories

After that, I evaluated accuracy of labeling images that does not included in the Caltech256. You can see the result in Table 2. I tested with 3 categories(Kangaroo, Duck and Dog) and 20 images each. I downloaded those 20 images in google that is similar to the dataset images(was for training) and I added some 1 or 2 images which is different in color or background with the dataset images. I could get the reasonable result of over than 90% accuracy. But the false is almost due to the difference of the color and background of images. This part needs to be improved.

4. CONCLUSION

In this paper, I design, implement and simply evaluate a system for image processing and image labeling. I designed this system to develop a simple approach to neural network for animal recognition in images. Since this process is implemented with the Caltech256 images, which is refined to have the rules and some kind of similarity, it has the limitation that I can't make sure that it would work for the images taken from the wild, which will have the different illuminations and different portion of animals. And also, as you can see the result from Table 2, it looks like it gives me out the false result when it label the images that is different(in color and in the color of background) from the trained images. I think this part needs to be improved.

But I can assure that I have worked for 2 months to build a system that someone who does not have the knowledge on the image classification at all can simply and easily work with the image classification in the field that needs to do it. In addition, just by following this process step by step, anyone can get the quick and reasonable result.

As a part of future work, I am investigating how to reduce the error rates occurred by the difference of the images by data augmentation to supplement the data shortage from the camera in wild.

5. REFERENCES

[1] Fei-Fei Li, Justin Johnson, 2016. CS231n lecture in Stanford university

[2] Tensorflow Org. TF-Slim code.

https://github.com/tensorflow/models/tree/master/slim