Dog Bread Identifier: Robotics Inference

Jung, Myoungki

Abstract—Inference has been on a big trend since neral network suggested new paradigm in compute vision. The accuracy and versitility of new trend, neural network influenced many different field of machine learning and allow developers create various filters at ease. This project shows how easy to make a inference filte from training a neural networks.

Index Terms—IEEEtran, Udacity, CNN, Inference.

1 Introduction

MODERN neural networks reignited machine learning provided many possibilities to robotics. Robotics inference is a key topic for robotics for operating tasks by a robot itself. Every task starts from recognising the surroundings first. Therefore, the accuracy of inference and latency of it is important for robotics field.

2 Background

There are two different neural networks were chosen to train the dataset. The required `P1_DATA` is a simple data structure and this could be solved by `AlexNet` and my custom dataset has more complicated and versatile features requires more sophisticated `GoogleLeNet` to be the trainer for the dataset. The motivation of my dataset is to classify the breed of the dog by infering a photo of dog.

3 Data Acquisition

3.1 Provided DATA

The `P1_DATA` was provided by Udacity and it has around 10 thousand photos of 3 categories.

3.2 Dog Bread Identifier

The `P1_DATA` was provided by Udacity and it has around 9 hundred photos of 12 categories. This smaller data reduces the resultant model smaller than others.

3.2.1 Dataset Download

The data set was scrapped from google image search. Using python script shown in the listing 1.

```
#!/usr/bin/python
from apiclient.discovery import build
import imghdr
import os
import sys
import urllib2
from google_images_download import
    google_images_download

orig_query = raw_input("Please input the text
    document path containing the desired image
    queries: ")
pet_type = orig_query[:-5]
query = orig_query.replace(' ', '_')
pets = [line.rstrip('\r\n') for line in open(query)]
type_pets =["\""+pet+"\"" for pet in pets]

args_pets = ','.join(type_pets)
arg_pets = ','.join(pets)

response = google_images_download.
    googleimagesdownload()   #class instantiation
arguments = {"keywords":args_pets,"format":"png","
    chromedriver":"/usr/lib/chromium-browser/
    chromedriver","limit":200,"size":"medium","
    aspect_ratio":"square","print_urls":False}   #
    creating list of arguments
paths = response.download(arguments)
print(paths)
```

Listing 1: Python script to download dataset

This script uses `chromedriver` to search, parse, download and save the google image search. There is a commercial google api for 1000 images download per five dollars. However, this script does the same function for free. Python package `google_images_download` is required to install prior to excute the script.

3.2.2 Dataset Preparation

Initially, 10 thousands for 130 different dog speices were prepared and it was reduced to data sample which is a portrait image of a dog with distinctive looks in order to help training. Noises in samples, multiple objects in the scene, unrelated objects in the scene, characters, and so on, did affect overall performance of network as these confuses the network back propagation functions and oscilates the loss values. Both udacity provided dataset and the my own dataset was divided to train, validation, test data set with ratio of 75:20:5. Table 1 shows the common details of both dataset, and the only difference is the `Dataset size`, 85.4 MB for my own dataset, and 930 MB.

```
#!/usr/bin/python
from PIL import Image
import os, sys, re
path = "~/Projects/Dataset/"
dirs = os.listdir( path )

def count_em(path):
      for root, dirs, files in sorted(os.walk(path))
    :
            for file_ in files:
            full_file_path = os.path.join(root,
    file_)
            print (full_file_path)
            try:
```

```
                    img = Image.open(full_file_path)
                    new_width  = 255
                    new_height = 255
                    img = img.resize((new_width,
    new_height), Image.ANTIALIAS)
                    img.save(os.path.join(root, file_+
    '').'png')
                except IOError, ex:
                    os.remove(full_file_path)

count_em(path)
```

Listing 2: Python script to resize and rename the file and directory name of dataset

After this step, archived data set was uploaded to udacity workspace and unzipped to create dataset database in DIGITS.

TABLE 1: Dataset in DIGITS

Parameter	Value
Image Dimensions	256x256 (Width x Height)
Image Type	Color
Resize Transformation	Squash
DB Backend	lmdb
Image Encoding	png
DB Compression	none

4 Hyperparameters

4.1 Training Hyperparameters

The difference is the larger number of training epochs, due to the smaller amount of dataset compared to the other. A policy to change the learning rate was applied to both training instaces as shown bottom of each Table. This helped to stablise the loss function.

4.1.1 *Udacity Dataset Training*

Training Hyperparameter was set in 'New Image Classification Model' section of DIGITS site. The Table 2 shows the used hyperparameters to train the `AlexNet` for the provided dataset. As the dataset has three categories and

TABLE 2: Details of Udacity Dataset Training Hyperparameters

Parameter	Value
Training epochs	30
Snapshot interval	every 1 epoch
Validation interval	every 1 epoch
Random seed	None
Batch size	None
Solver type	AdaGrad
Base Learning Rate	0.005
Solver type	AdaGrad
Policy	stepdown
Step Size	0.33
Gamma	0.5
Network	AlexNet

the amount of data is large only 30 epochs could meet the project requirement, an inference accuracy more than 75 percents.

(a) First 15 Epoches of training with starting learning rate 0.005

(b) Second 15 Epoches with starting learning rate 0.00125

Fig. 1: Analysis for run4

4.1.2 *Dog Breed Dataset Training*

The hyperparameters for GoogleLeNet network for dog breed identifier network was set as Table 3.

TABLE 3: Details of Dog Breed Dataset Training Hyperparameters

Parameter	Value
Training epochs	75
Snapshot interval	every 1 epoch
Validation interval	every 1 epoch
Random seed	None
Batch size	None
Solver type	AdaGrad
Base Learning Rate	0.005
Solver type	AdaGrad
Policy	stepdown
Step Size	0.33
Gamma	0.5
Network	GoogLeNet [1]

5 Results

5.1 AlexNet for Udacity Dataset

Figure 1 shows the enitre training progressof the AlexNet for provided dataset. Figure 2 indicates that the result for evaluation using 'evaluate' command on a terminal passes all the requirements for the project. The evaluation result, 5 ms inference and 75.3 percent accuracy, on Figure 2 exceeds the project requirement, under 10ms inference, and 75 percent accuracy.

5.2 Dog Breed Identifier Network

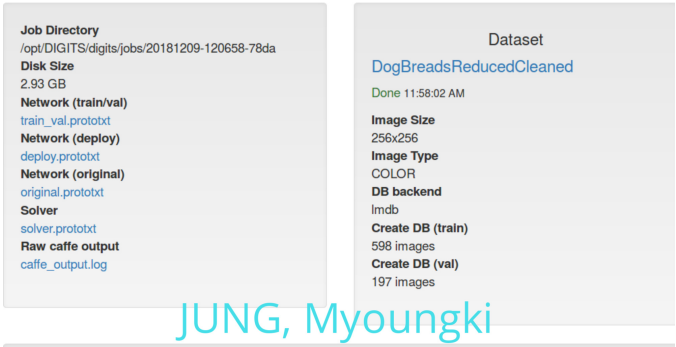
The first overall 80 percent of touching the tube with gripper is shown on Figure 3.

Because there is no automated script to test the performance of the trained network, the inference accuracy with

```
root@02cb4e097308:/home/workspace# evaluate

Do not run while you are processing data or training a model.

Please enter the Job ID: 20181209-143859-8ecf

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20181209-143859-8ecf/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20181209-143859-8ecf/snapshot_iter_900.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x227x227
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 4.40439 ms.
Average over 10 runs is 4.39527 ms.
Average over 10 runs is 4.38442 ms.
Average over 10 runs is 4.37996 ms.
Average over 10 runs is 4.12608 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14652  100 12336  100  2316    938    176  0:00:13  0:00:13 --:--:--  2427

Your model accuacy is 75.4098360656 %
root@02cb4e097308:/home/workspace#
```

JUNG, Myoungki

Fig. 2: Agent reaching reaching with gripper



JUNG, Myoungki

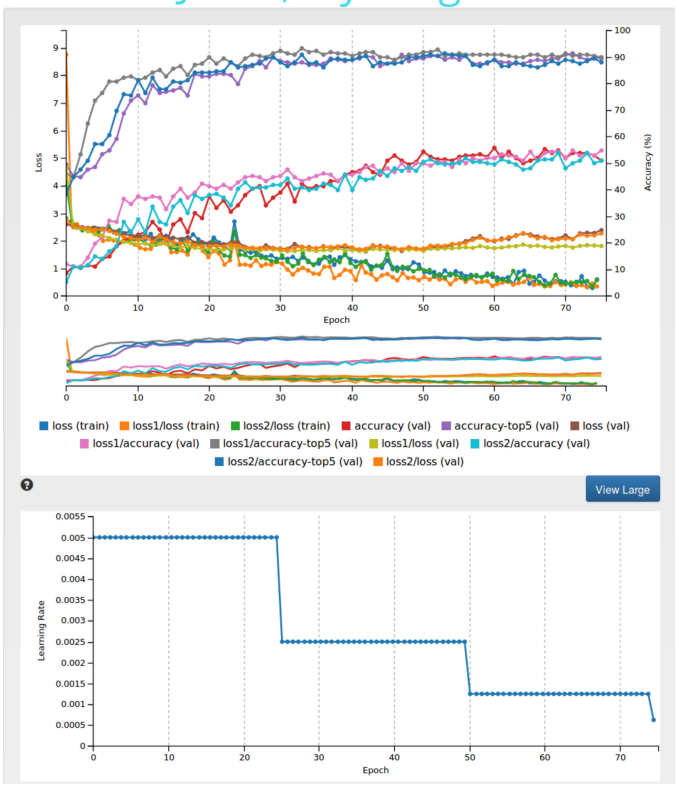
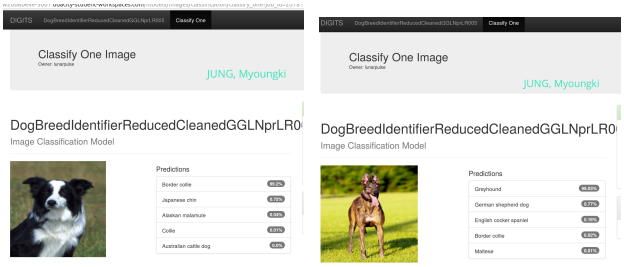
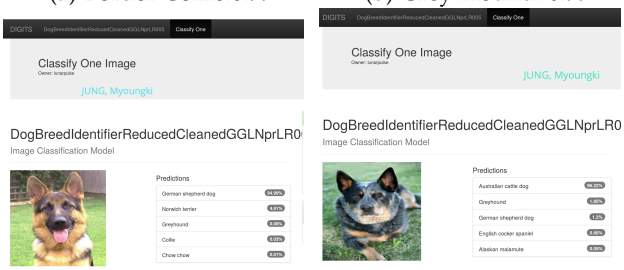


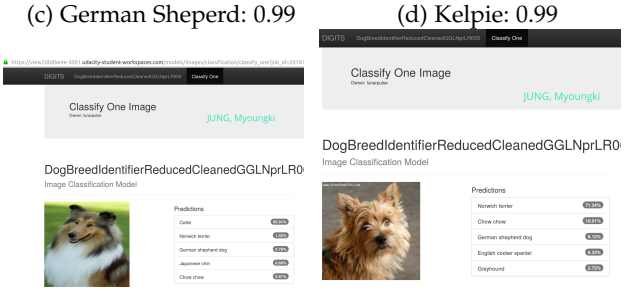
Fig. 3: Result of Dog Breed Identifier Network Training



(a) Border Collie:0.99 (b) Grey Hound: 0.99



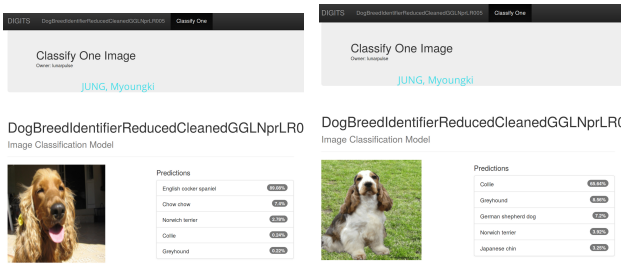
(c) German Sheperd: 0.99 (d) Kelpie: 0.99



(e) Collie: 0.95 (f) Norwich Terrier: 0.71

Fig. 4: High accuracy of inference from the trained network

manual random inference check shows the high accuracy of trained network, Figure 4 and poor inference results 5.



(a) Understand but not sure (b) Not even listed

Fig. 5: Poor accuracy of inference from the trained network

The project related files listed below are archived into tar.gz or zip files.

- `deploy.prototxt`
- `labels.txt`
- `mean.binaryproto`
- `your_model.caffemodel`
- `solver.prototxt`
- `train_val.prototxt`

Dog breed identifier network related are archived as:

- `DogBreedIdentifierReducedCleanedGGLNprLR005/20181209-120658-78da_epoch_75.0.tar.gz` : contains the epoch 51 model configuration for dog breed inference network.
- `DogBreedIdentifierReducedCleanedGGLNprLR005/20181209-120658-78da_epoch_75.0.tar.gz` : contains the epoch 51 model configuration for dog breed inference network.
- `DogBreedIdentifierReducedCleanedGGLNprLR005/dogImagesReduced.zip` : Reduced Dataset (only 12 categories)

The archive files related to the classification network for Udacity provided data, `P1_DATA`, are:

- `providedModel75p/20181209-143859-8ecf_epoch_15.0.tar.gz` : contains the epoch 30 of model configuration for the classification network for Udacity provided data.

6 Discussion

Obtaining approperiate dataset was the hardest start. The dataset must be diverse and not containing too many unrelated objects and the larger amount of it is better training result. Poor inference also inherits from the biased data, or not well prepared data. Some postures of the dogs in the datasample is rather fewer than other posture and these rare samples limits the accuracy if a trained network is tested with such less experienced test dataset. In other words, dataset for training should contain as many cases in variety and amount as possible. Training the neural networks for the models took a great deal of attention. The networks were well pre-defined by default, however, the training epochs and learning rate affected its stability and accuracy significantly. From many trials, learning rate 0.005 was a good starting point which lifts the training accuracy of the network to 90 percent within 10 epochs and learning rate lower than that value resulted in low overall training accuracy by not applying the loss to the network well, especially with the policy of reduction of a learning rate. With high value of learning rate more than 0.075, the loss and accuracy oscilates in later epochs did not reach high accuracy after a training.

7 Conclusion / Future work

The project The inference using name networks performs with an adequate dataset and training. The complicated convolutional networks perform well in many cases, still many other new architectures are introduced. CapsNet [2] utilises dynamic routing algorithm outperforms CNN in terms of accuracy and inference completeness for entire layout of the inference images. The framework usded in the DIGITS is caffee and its syntax is different from well known frameworks such as `tensorflow`. If porting to online inference on a robot is considered, `Tensorflow RT` is suggested as NVIDIA Jetson embraces this framework to utilise their CUDA processors optimally. Inference tasks in an active research field and there will be more better networks coming and the developers have to keep watching the trend to implment the greatness of the techonology.

References

[1] C. Szegedy, V. Vanhoucke, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2014.
[2] S. Sabour, C. V. Nov, and G. E. Hinton, “Dynamic Routing Between Capsules,” no. Nips, 2017.