Udacity Term 2 Robotic Inference Project

Abstract

This project has two classification data models using Deep Neural Network technology. The first one is classifying the bottles, candy wrappers and nothing on a moving belt. The second one classified the dog A, dog B or nothing. The image data are come from public dataset. The project used the three model: AlexNet, GoogLeNet and LeNet. For both classifications, the best results were presented in this article.

Introduction

Classification includes a broad range of decision-theoretic approaches to the identification of images (or parts thereof). All classification algorithms are based on the assumption that the image in question depicts one or more features (*e.g.*, geometric parts in the case of a manufacturing classification system, or spectral regions in the case of remote sensing) and that each of these features belongs to one of several distinct and exclusive classes. The classes may be specified *a priori* by an analyst (as in *supervised classification*) or automatically clustered (*i.e.* as in *unsupervised classification*) into sets of prototype classes, where the analyst merely specifies the number of desired categories[1].

In this project used NVIDIA's DIGITS workflow[2] to rapidly prototype ideas that can be deployed on the Jetson in close to real time. The DIGITS will prototype classification networks, detection networks, segmentation networks!

There are two parts in the project:

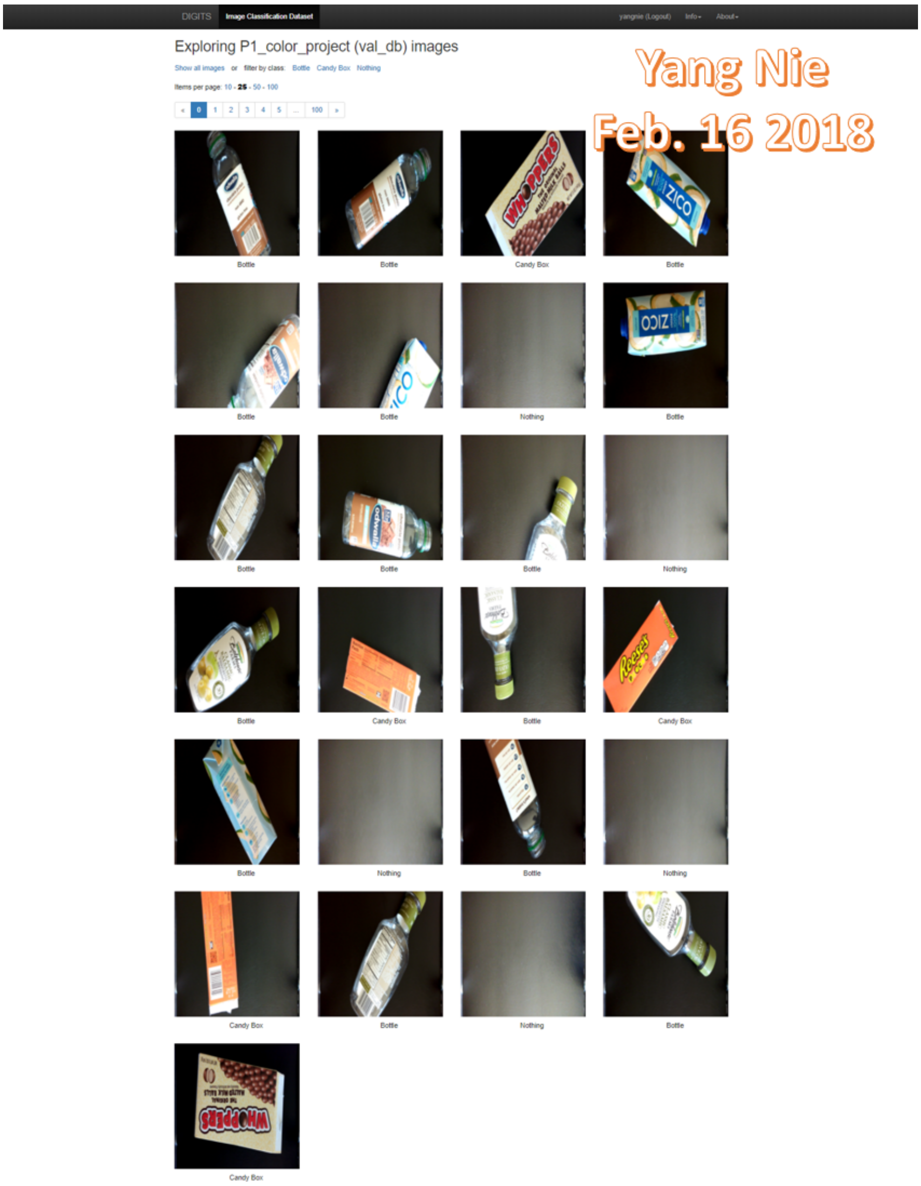
1. P1 moving belt image classification part used P1 dataset pictures of candy boxes, bottles, and nothing (empty conveyor belt).
2. Dog image classification part used the dog image dataset (dog A, dog B and nothing) which Author collected from iPhone.

Background / Formulation

1. P1 moving belt image classification

The P1 image dataset is stored in /data/P1/ directory. It include all images of bottles, candy wrappers and no object on a conveyor belt passing under a camera. A swing arm is used to sort all right objects to correct the bins depending on classifying results.

P1 dataset image example:



Exploring P1_color_project (val_db) images

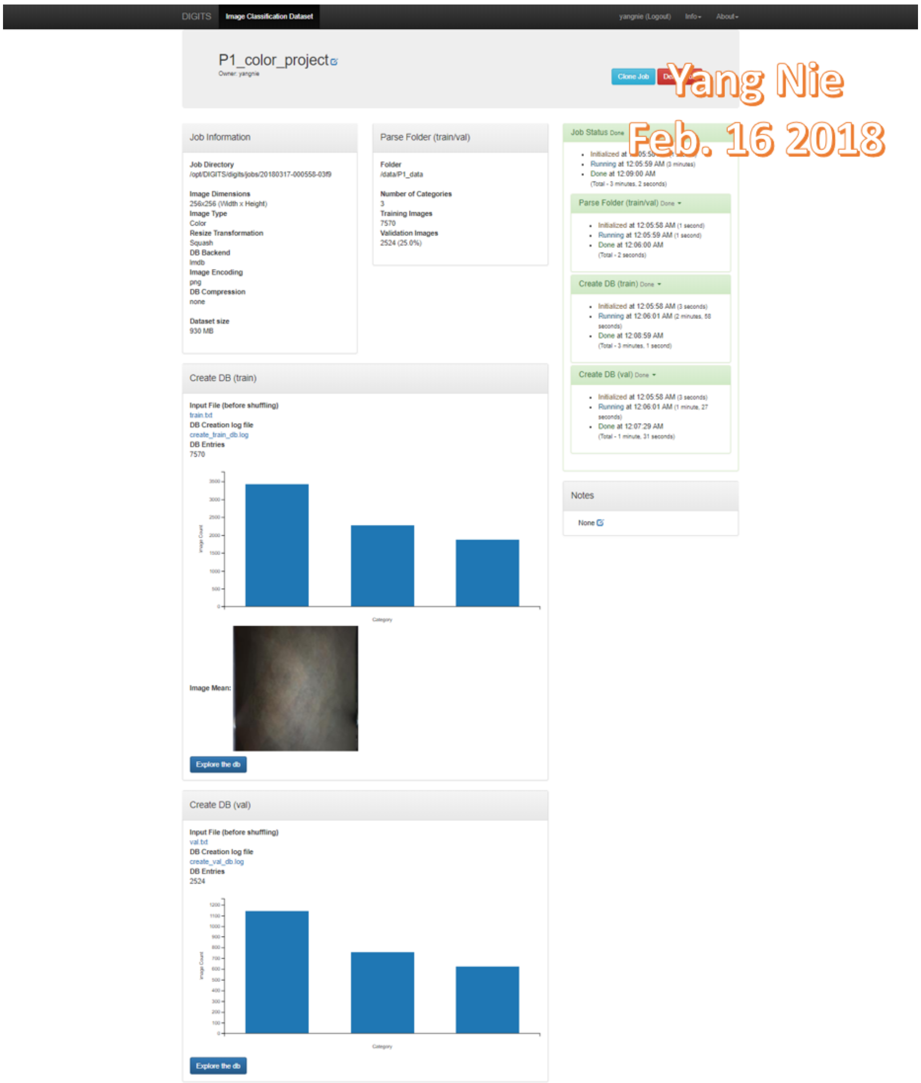
Show all images or filter by class: Bottle Candy Box Nothing

Items per page: 10 - **25** - 50 - 100

Yang Nie

Feb. 16 2018

P1 dataset was split two training and validation parts, the color image size is 256 X 256



Yang Nie
Feb. 16 2018

P1_color_project
Owner: yangnie

Job Information

Job Directory
/opt/DIGITS/digits/jobs/20180317-000558-03f9

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

Dataset size
930 MB

Parse Folder (train/val)

Folder
/data/P1_data

Number of Categories
3
Training Images
7570
Validation Images
2524 (25.0%)

Job Status Done

- Initialized at [illegible]05:58 [illegible]
- Running at 12:05:59 AM (3 minutes)
- Done at 12:09:00 AM
  (Total - 3 minutes, 2 seconds)

Parse Folder (train/val) Done

- Initialized at 12:05:58 AM (1 second)
- Running at 12:05:59 AM (1 second)
- Done at 12:06:00 AM
  (Total - 2 seconds)

Create DB (train) Done

- Initialized at 12:05:58 AM (3 seconds)
- Running at 12:06:01 AM (2 minutes, 58 seconds)
- Done at 12:08:59 AM
  (Total - 3 minutes, 1 second)

Create DB (val) Done

- Initialized at 12:05:58 AM (3 seconds)
- Running at 12:06:01 AM (1 minute, 27 seconds)
- Done at 12:07:29 AM
  (Total - 1 minute, 31 seconds)

Notes

None

Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log
DB Entries
7570

Image Mean:

Create DB (val)

Input File (before shuffling)
val.txt
DB Creation log file
create_val_db.log
DB Entries
2524

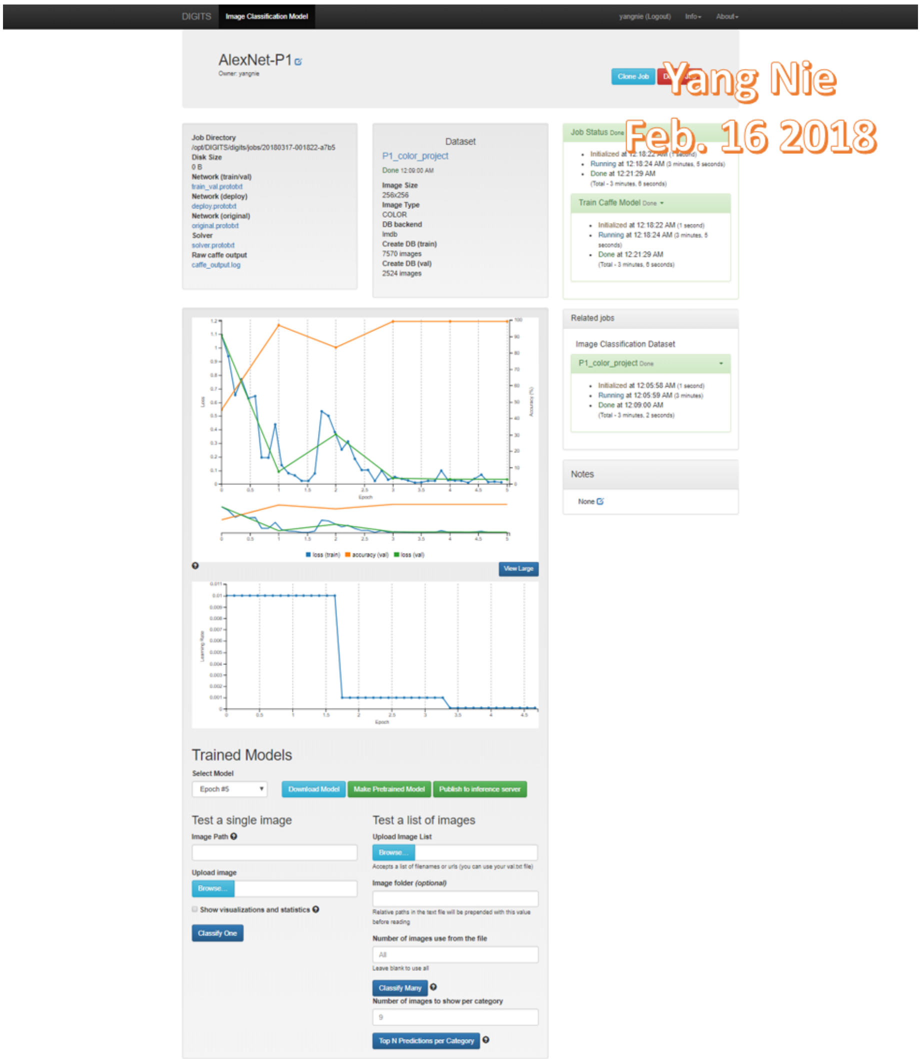
Data Acquisition:

The P1 dataset is provided from Udacity Robotics lesson.

It were split to two sets: training and validation dataset. They are color image and size is 256 x 256. This dataset is provided from Udacity robotics class.

Model creation:

AlexNet Model was built as:



Yang Nie
Feb. 16 2018

AlexNet-P1

Owner: yangnie

Job Directory
/opt/DIGITS/digits/jobs/20180317-001822-a7b5
Disk Size
0 B
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Dataset
P1_color_project
Done 12:09:00 AM
Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
7570 images
Create DB (val)
2524 images

Job Status Done
- Initialized at 12:18:22 AM (1 second)
- Running at 12:18:24 AM (3 minutes, 5 seconds)
- Done at 12:21:29 AM

(Total - 3 minutes, 6 seconds)

Train Caffe Model Done
- Initialized at 12:18:22 AM (1 second)
- Running at 12:18:24 AM (3 minutes, 5 seconds)
- Done at 12:21:29 AM

(Total - 3 minutes, 6 seconds)

Related jobs

Image Classification Dataset

P1_color_project Done
- Initialized at 12:05:58 AM (1 second)
- Running at 12:05:59 AM (3 minutes)
- Done at 12:09:00 AM

(Total - 3 minutes, 2 seconds)

Notes

None

Trained Models

Select Model
Epoch #5

Test a single image

Image Path

Upload image

□ Show visualizations and statistics

Test a list of images

Upload Image List

Accepts a list of filenames or urls (you can use your val.txt file)

Image folder *(optional)*

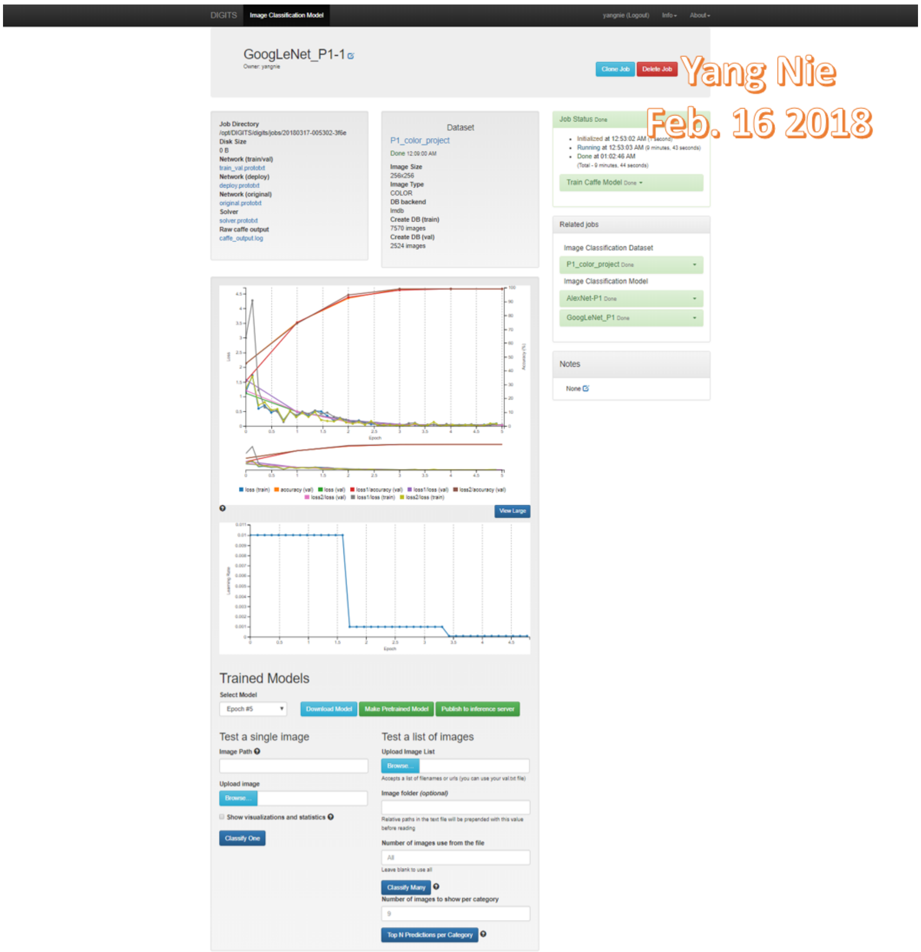
Relative paths in the text file will be prepended with this value before reading

Number of images use from the file
All

Leave blank to use all

Number of images to show per category
9

GoogLeNet Model was built as:



Yang Nie

Feb. 16 2018

GoogLeNet_P1-1

Owner: yangnie

Job Directory
/opt/DIGITS/digits/jobs/20180317-005302-3f6e
Disk Size
0 B
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Dataset
P1_color_project
Done 12:09:00 AM
Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
7570 images
Create DB (val)
2524 images

Job Status Done

- Initialized at 12:53:02 AM (1 second)
- Running at 12:53:03 AM (9 minutes, 43 seconds)
- Done at 01:02:46 AM
  (Total - 9 minutes, 44 seconds)

Train Caffe Model Done

Related jobs

Image Classification Dataset
P1_color_project Done
Image Classification Model
AlexNet-P1 Done
GoogLeNet_P1 Done

Notes

None

Trained Models

Select Model
Epoch #5

Test a single image

Image Path

Upload image

□ Show visualizations and statistics

Test a list of images

Upload Image List

Accepts a list of filenames or urls (you can use your val.txt file)

Image folder *(optional)*

Relative paths in the text file will be prepended with this value before reading

Number of images use from the file
All
Leave blank to use all

Number of images to show per category
9

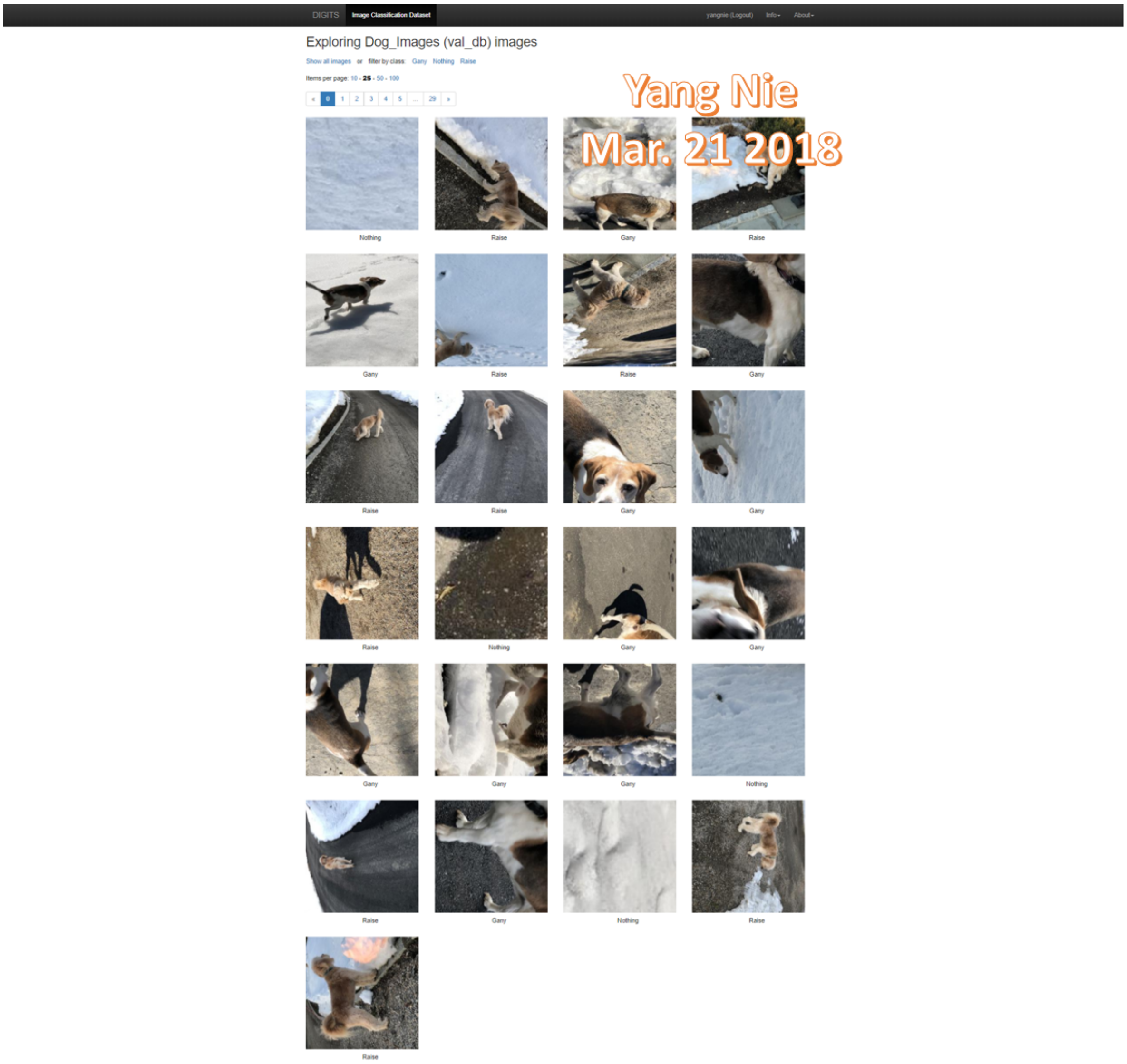
The parameter setting:

Both epoch were set to 5. All other parameters used as default.

2. Dog image classification

The dog image files located in Output folder, it includes three subfolders, Gany for dog A, Raise for dog B, Nothing for no object.

Dog image example:



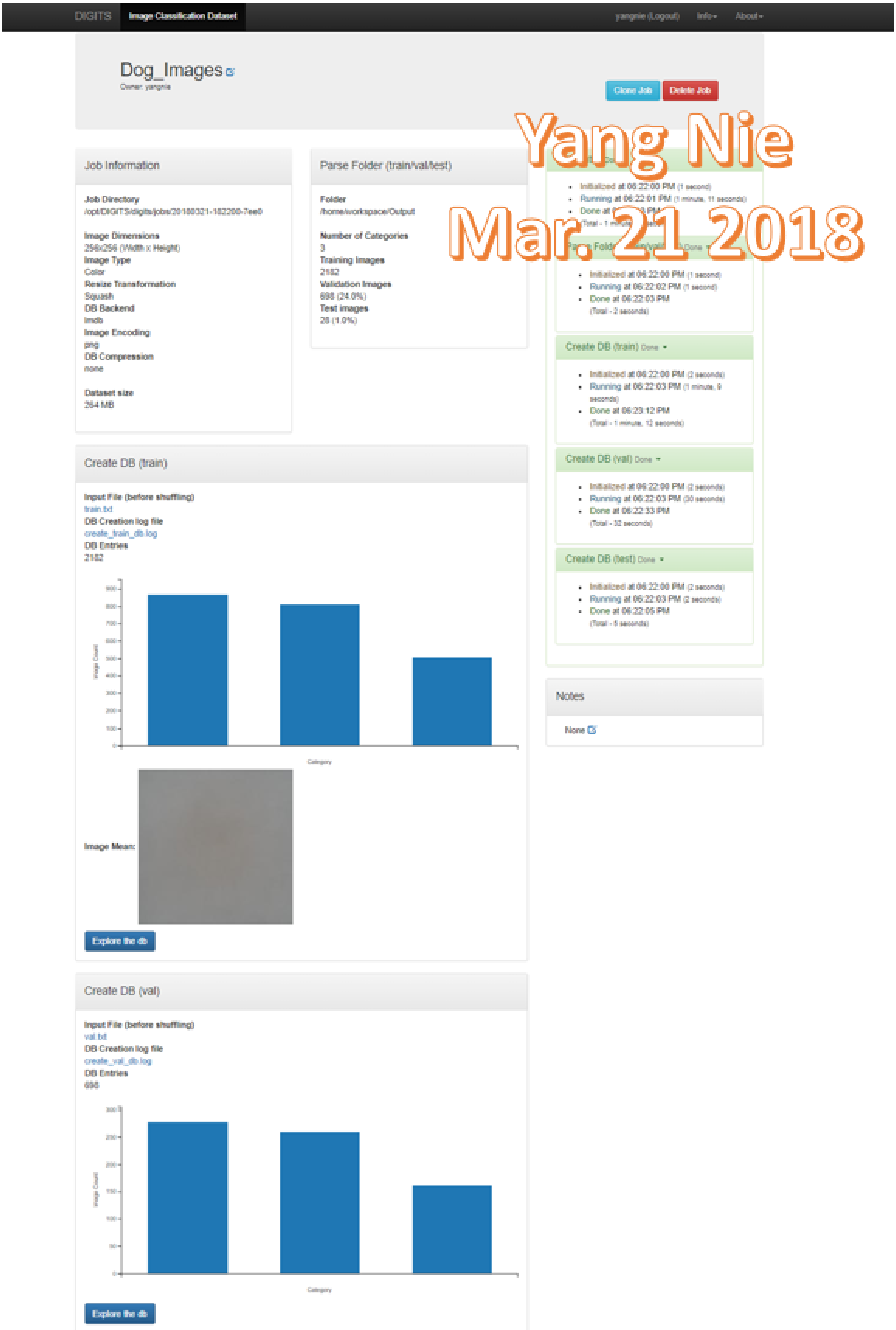
Exploring Dog_Images (val_db) images

Show all images or filter by class: Gany Nothing Raise

Items per page: 10 - 25 - 50 - 100

Yang Nie
Mar. 21 2018

Dog image dataset was split three training, validation and test parts, the color image size is 256 X 256



Yang Nie

Mar. 21 2018

Dog_Images

Owner: yangnie

Job Information

Job Directory
/opt/DIGITS/digits/jobs/20180321-182200-7ee0

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

Dataset size
264 MB

Parse Folder (train/val/test)

Folder
/home/workspace/Output

Number of Categories
3
Training Images
2182
Validation Images
698 (24.0%)
Test images
28 (1.0%)

- Initialized at 06:22:00 PM (1 second)
- Running at 06:22:01 PM (1 minute, 11 seconds)
- Done at 0 PM

(Total - 1 minute)

Parse Folder (train/val/test) Done ▾

- Initialized at 06:22:00 PM (1 second)
- Running at 06:22:02 PM (1 second)
- Done at 06:22:03 PM

(Total - 2 seconds)

Create DB (train) Done ▾

- Initialized at 06:22:00 PM (2 seconds)
- Running at 06:22:03 PM (1 minute, 9 seconds)
- Done at 06:23:12 PM

(Total - 1 minute, 12 seconds)

Create DB (val) Done ▾

- Initialized at 06:22:00 PM (2 seconds)
- Running at 06:22:03 PM (30 seconds)
- Done at 06:22:33 PM

(Total - 32 seconds)

Create DB (test) Done ▾

- Initialized at 06:22:00 PM (2 seconds)
- Running at 06:22:03 PM (2 seconds)
- Done at 06:22:05 PM

(Total - 5 seconds)

Notes

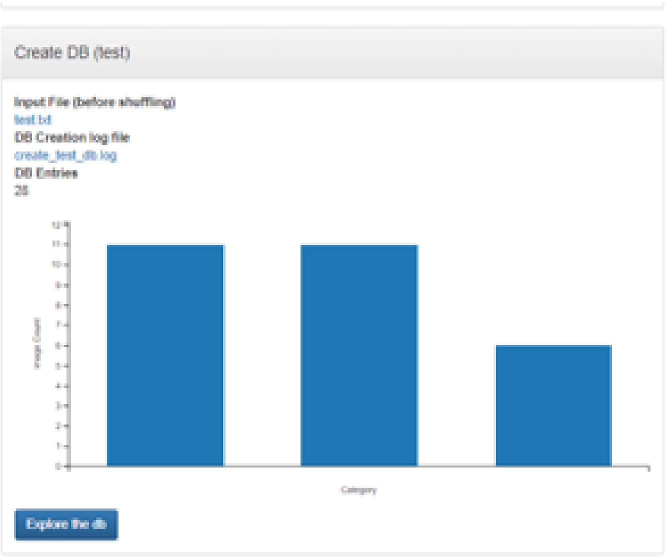
None

Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log
DB Entries
2182

Create DB (val)

Input File (before shuffling)
val.txt
DB Creation log file
create_val_db.log
DB Entries
698



Create DB (test)

Input File (before shuffling)
test.txt
DB Creation log file
create_test_db.log
DB Entries
28

Data Acquisition:

The dog images were taken from iPhone, then used Augmentation[3] code to generate 500 to 1000 additional images depend the object type.

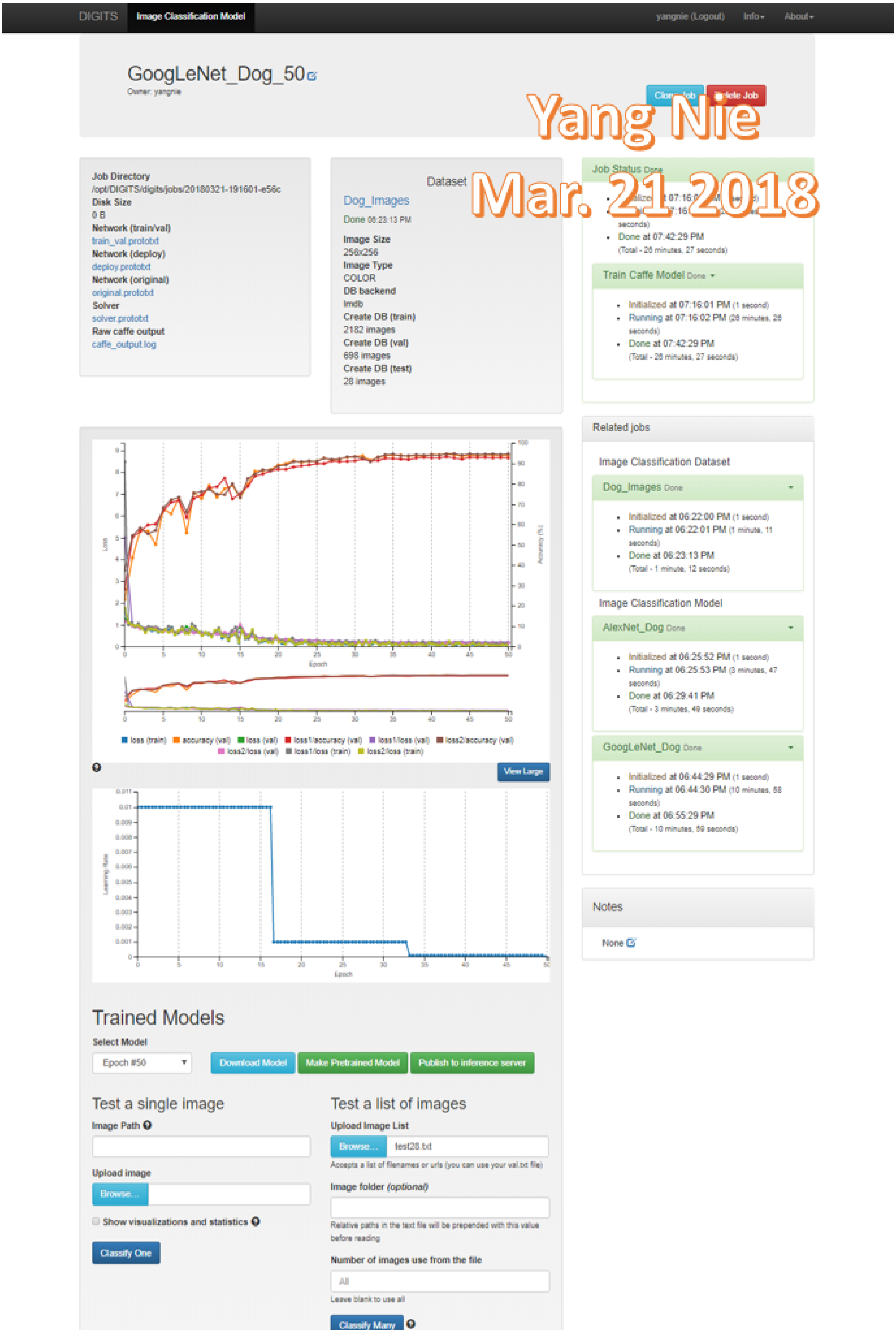
	Dog A (Dany)	Dog B (Raise)	No object (Nothing)
iPhone images	193	156	62
Augment generated images	887	998	612
Total images	1080	1154	674

The dog database was created as:

	Training	Validating	Testing
Image number	2185	695	28
Percentage	75%	24%	1%

Model creation:

GoogLeNet Model was created as : (Use default GoogLeNet network, no change in model itself)



DIGITS

Image Classification Model

yangnie (Logout)

Info▾

About▾

GoogLeNet_Dog_50

Owner: yangnie

Clone Job

Delete Job

Yang Nie

Mar. 21 2018

Job Directory
/opt/DIGITS/digits/jobs/20180321-191601-e56c
Disk Size
0 B
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Dataset

Dog_Images

Done 06:23:13 PM

Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
2182 images
Create DB (val)
698 images
Create DB (test)
28 images

Job Status Done

- Initialized at 07:16:01 PM (1 second)
- Running at 07:16:02 PM (26 minutes, 26 seconds)
- Done at 07:42:29 PM
  (Total - 26 minutes, 27 seconds)

Train Caffe Model Done ▾

- Initialized at 07:16:01 PM (1 second)
- Running at 07:16:02 PM (26 minutes, 26 seconds)
- Done at 07:42:29 PM
  (Total - 26 minutes, 27 seconds)

Loss
Accuracy (%)
Epoch
0
5
10
15
20
25
30
35
40
45
50
■ loss (train) ■ accuracy (val) ■ loss (val) ■ loss1/accuracy (val) ■ loss1/loss (val) ■ loss2/accuracy (val) ■ loss2/loss (val) ■ loss1/loss (train) ■ loss2/loss (train)
View Large
Learning Rate
0.011
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0

Related jobs

Image Classification Dataset

Dog_Images Done ▾

- Initialized at 06:22:00 PM (1 second)
- Running at 06:22:01 PM (1 minute, 11 seconds)
- Done at 06:23:13 PM
  (Total - 1 minute, 12 seconds)

Image Classification Model

AlexNet_Dog Done ▾

- Initialized at 06:25:52 PM (1 second)
- Running at 06:25:53 PM (3 minutes, 47 seconds)
- Done at 06:29:41 PM
  (Total - 3 minutes, 49 seconds)

GoogLeNet_Dog Done ▾

- Initialized at 06:44:29 PM (1 second)
- Running at 06:44:30 PM (10 minutes, 58 seconds)
- Done at 06:55:29 PM
  (Total - 10 minutes, 59 seconds)

Notes

None

Trained Models

Select Model

Epoch #50 ▾

Download Model

Make Pretrained Model

Publish to inference server

Test a single image

Image Path

Upload image

Browse...

□ **Show visualizations and statistics**

Classify One

Test a list of images

Upload Image List

Browse... test28.txt

Accepts a list of filenames or urls (you can use your val.txt file)

Image folder *(optional)*

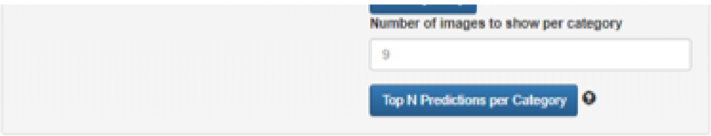
Relative paths in the text file will be prepended with this value before reading

Number of images use from the file

All

Leave blank to use all

Classify Many



Number of images to show per category

9

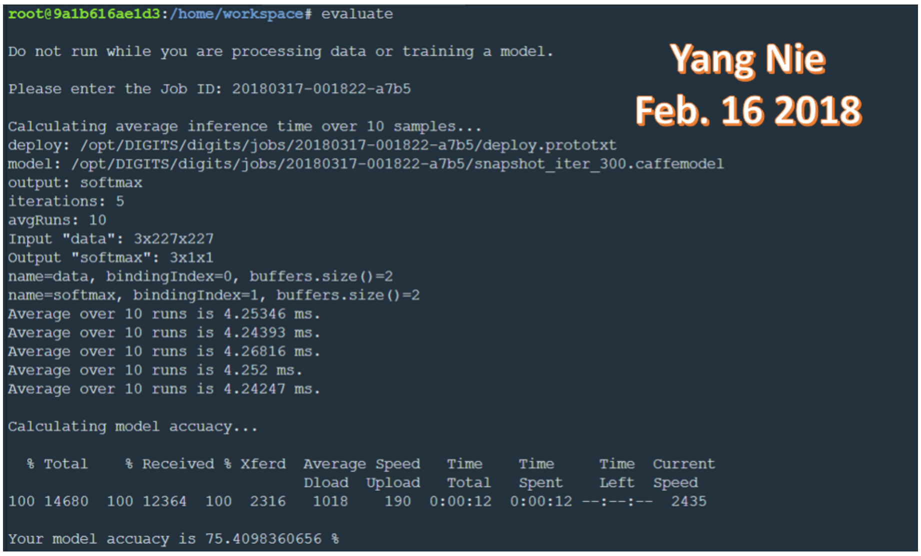
The parameter setting:

Both epoch were set to 50 and Batch size = 50. All other parameters used as default.

Results

1. P1 moving belt image classification

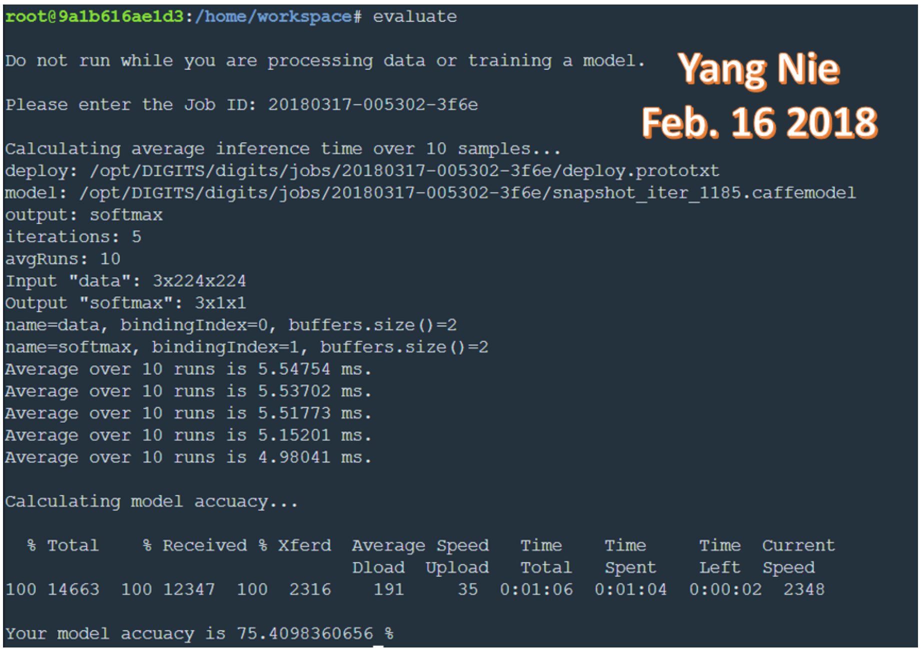
Evaluating result for AlexNet Model as:



Yang Nie
Feb. 16 2018

```
root@9a1b616ae1d3:/home/workspace# evaluate

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

Please enter the Job ID: 20180317-001822-a7b5

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180317-001822-a7b5/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180317-001822-a7b5/snapshot_iter_300.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 4.25346 ms.
Average over 10 runs is 4.24393 ms.
Average over 10 runs is 4.26816 ms.
Average over 10 runs is 4.252 ms.
Average over 10 runs is 4.24247 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14680  100 12364  100  2316   1018    190  0:00:12  0:00:12 --:--:--  2435

Your model accuacy is 75.4098360656 %
```

Evaluating result for GoogLeNet Model as:



Yang Nie
Feb. 16 2018

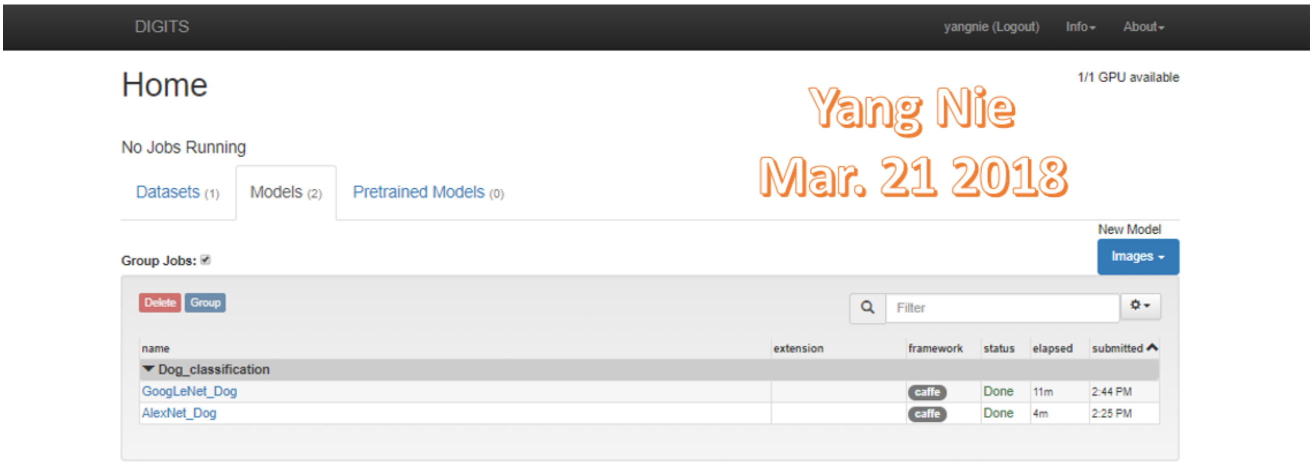
```
root@9a1b616ae1d3:/home/workspace# evaluate

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

Please enter the Job ID: 20180317-005302-3f6e

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180317-005302-3f6e/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180317-005302-3f6e/snapshot_iter_1185.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.54754 ms.
Average over 10 runs is 5.53702 ms.
Average over 10 runs is 5.51773 ms.
Average over 10 runs is 5.15201 ms.
Average over 10 runs is 4.98041 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14663  100 12347  100  2316    191     35  0:01:06  0:01:04  0:00:02  2348

Your model accuacy is 75.4098360656 %
```

Both AlexNet and GoogLeNet models are at least 75 percent accuracy and an inference time of less than 10 ms.

	AlexNet	**GoogLeNet**
Accuacy	75.4090360656%	75.4090360656%
Average inference Time	4.254004 ms	5.34768 ms

2. Dog image classification

AlexNet and GoogLeNet models, both were built and tested. Compare both of them, the GoogLeNet model has better results. This article only shows GoogLeNet model result.



Home

1/1 GPU available

Yang Nie

Mar. 21 2018

No Jobs Running

Group Jobs: ☑

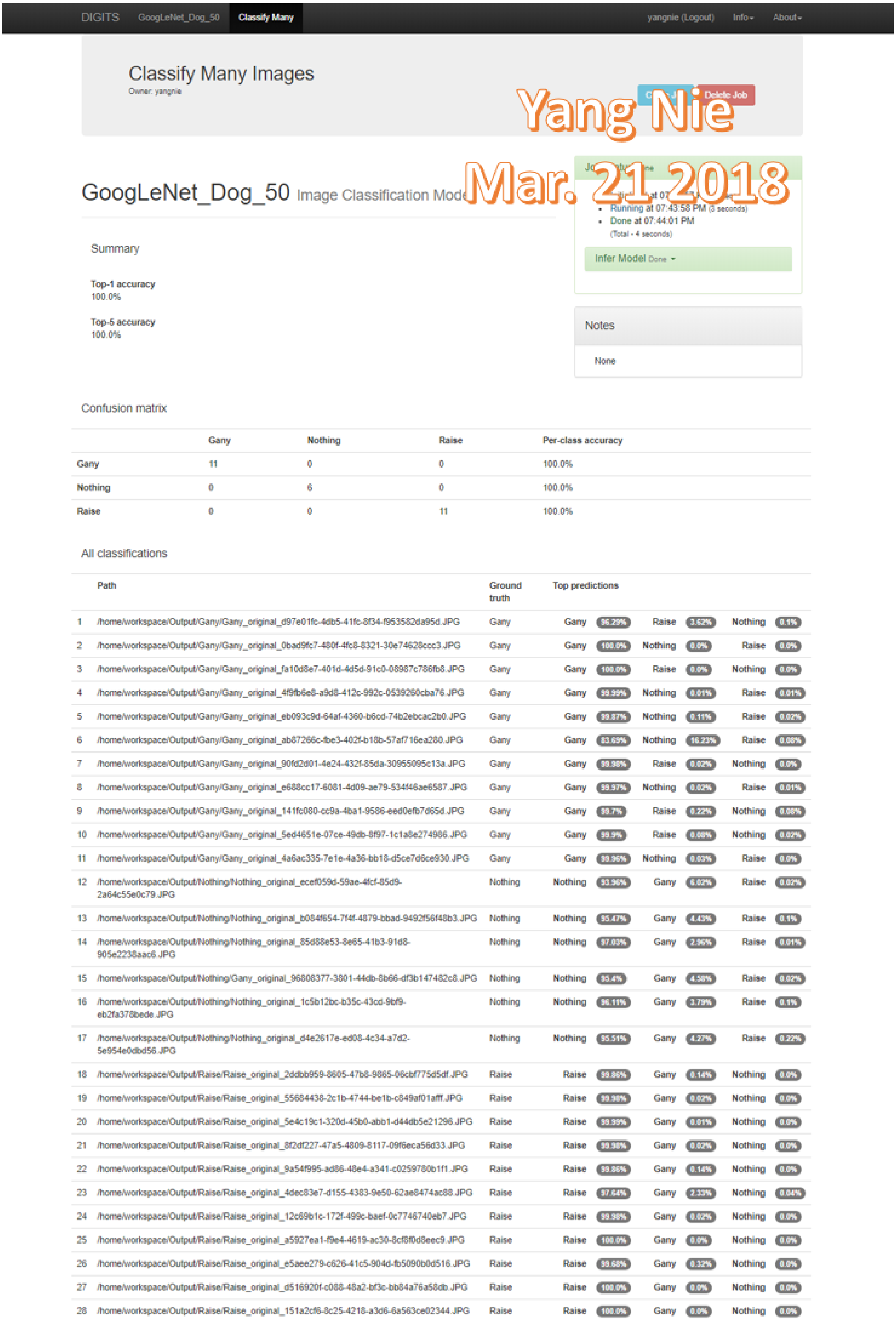
name	extension	framework	status	elapsed	submitted
Dog_classification					
GoogLeNet_Dog		caffe	Done	11m	2:44 PM
AlexNet_Dog		caffe	Done	4m	2:25 PM

This is GoogLeNet model results (epoch = 50, batch size = 50): The result is very good, the accuracy for all three classes are 100%.

Note: The number in table is the number of image:

	Dog A (Gany)	**Dog B (Raise)**	**No object (Nothing)**	**Per-class accuracy**
Dog A (Gany)	11	0	0	100%
Dog B (Raise)	0	11	0	100%
No object (Nothing)	0	0	6	100%

Digits test results screen copy:



Yang Nie

Mar. 21 2018

Classify Many Images

Owner: yangnie

- Running at 07:43:58 PM (3 seconds)
- Done at 07:44:01 PM
  (Total - 4 seconds)

GoogLeNet_Dog_50 Image Classification Mode

Summary

Top-1 accuracy
100.0%

Top-5 accuracy
100.0%

Notes

None

Confusion matrix

	Gany	Nothing	Raise	Per-class accuracy
Gany	11	0	0	100.0%
Nothing	0	6	0	100.0%
Raise	0	0	11	100.0%

All classifications

	Path	Ground truth	Top predictions					
1	/home/workspace/Output/Gany/Gany_original_d97e01fc-4db5-41fc-8f34-f953582da95d.JPG	Gany	Gany	96.29%	Raise	3.62%	Nothing	0.1%
2	/home/workspace/Output/Gany/Gany_original_0bad9fc7-480f-4fc8-8321-30e74628ccc3.JPG	Gany	Gany	100.0%	Nothing	0.0%	Raise	0.0%
3	/home/workspace/Output/Gany/Gany_original_fa10d8e7-401d-4d5d-91c0-08987c786fb8.JPG	Gany	Gany	100.0%	Raise	0.0%	Nothing	0.0%
4	/home/workspace/Output/Gany/Gany_original_4f9fb6e8-a9d8-412c-992c-0539260cba76.JPG	Gany	Gany	99.99%	Nothing	0.01%	Raise	0.01%
5	/home/workspace/Output/Gany/Gany_original_eb093c9d-64af-4360-b6cd-74b2ebcac2b0.JPG	Gany	Gany	99.87%	Nothing	0.11%	Raise	0.02%
6	/home/workspace/Output/Gany/Gany_original_ab87266c-fbe3-402f-b18b-57af716ea280.JPG	Gany	Gany	83.69%	Nothing	16.23%	Raise	0.08%
7	/home/workspace/Output/Gany/Gany_original_90fd2d01-4e24-432f-85da-30955095c13a.JPG	Gany	Gany	99.98%	Raise	0.02%	Nothing	0.0%
8	/home/workspace/Output/Gany/Gany_original_e688cc17-6081-4d09-ae79-534f46ae6587.JPG	Gany	Gany	99.97%	Nothing	0.02%	Raise	0.01%
9	/home/workspace/Output/Gany/Gany_original_141fc080-cc9a-4ba1-9586-eed0efb7d65d.JPG	Gany	Gany	99.7%	Raise	0.22%	Nothing	0.08%
10	/home/workspace/Output/Gany/Gany_original_5ed4651e-07ce-49db-8f97-1c1a8e274986.JPG	Gany	Gany	99.9%	Raise	0.08%	Nothing	0.02%
11	/home/workspace/Output/Gany/Gany_original_4a6ac335-7e1e-4a36-bb18-d5ce7d6ce930.JPG	Gany	Gany	99.96%	Nothing	0.03%	Raise	0.0%
12	/home/workspace/Output/Nothing/Nothing_original_ecef059d-59ae-4fcf-85d9-2a64c55e0c79.JPG	Nothing	Nothing	93.96%	Gany	6.02%	Raise	0.02%
13	/home/workspace/Output/Nothing/Nothing_original_b084f654-7f4f-4879-bbad-9492f56f48b3.JPG	Nothing	Nothing	95.47%	Gany	4.43%	Raise	0.1%
14	/home/workspace/Output/Nothing/Nothing_original_85d88e53-8e65-41b3-91d8-905e2238aac6.JPG	Nothing	Nothing	97.03%	Gany	2.96%	Raise	0.01%
15	/home/workspace/Output/Nothing/Gany_original_96808377-3801-44db-8b66-df3b147482c8.JPG	Nothing	Nothing	95.4%	Gany	4.58%	Raise	0.02%
16	/home/workspace/Output/Nothing/Nothing_original_1c5b12bc-b35c-43cd-9bf9-eb2fa378bede.JPG	Nothing	Nothing	96.11%	Gany	3.79%	Raise	0.1%
17	/home/workspace/Output/Nothing/Nothing_original_d4e2617e-ed08-4c34-a7d2-5e954e0dbd56.JPG	Nothing	Nothing	95.51%	Gany	4.27%	Raise	0.22%
18	/home/workspace/Output/Raise/Raise_original_2ddbb959-8605-47b8-9865-06cbf775d5df.JPG	Raise	Raise	99.86%	Gany	0.14%	Nothing	0.0%
19	/home/workspace/Output/Raise/Raise_original_55684438-2c1b-4744-be1b-c849af01afff.JPG	Raise	Raise	99.98%	Gany	0.02%	Nothing	0.0%
20	/home/workspace/Output/Raise/Raise_original_5e4c19c1-320d-45b0-abb1-d44db5e21296.JPG	Raise	Raise	99.99%	Gany	0.01%	Nothing	0.0%
21	/home/workspace/Output/Raise/Raise_original_8f2df227-47a5-4809-8117-09f6eca56d33.JPG	Raise	Raise	99.98%	Gany	0.02%	Nothing	0.0%
22	/home/workspace/Output/Raise/Raise_original_9a54f995-ad86-48e4-a341-c0259780b1f1.JPG	Raise	Raise	99.86%	Gany	0.14%	Nothing	0.0%
23	/home/workspace/Output/Raise/Raise_original_4dec83e7-d155-4383-9e50-62ae8474ac88.JPG	Raise	Raise	97.64%	Gany	2.33%	Nothing	0.04%
24	/home/workspace/Output/Raise/Raise_original_12c69b1c-172f-499c-baef-0c7746740eb7.JPG	Raise	Raise	99.98%	Gany	0.02%	Nothing	0.0%
25	/home/workspace/Output/Raise/Raise_original_a5927ea1-f9e4-4619-ac30-8cf8f0d8eec9.JPG	Raise	Raise	100.0%	Gany	0.0%	Nothing	0.0%
26	/home/workspace/Output/Raise/Raise_original_e5aee279-c626-41c5-904d-fb5090b0d516.JPG	Raise	Raise	99.68%	Gany	0.32%	Nothing	0.0%
27	/home/workspace/Output/Raise/Raise_original_d516920f-c088-48a2-bf3c-bb84a76a58db.JPG	Raise	Raise	100.0%	Gany	0.0%	Nothing	0.0%
28	/home/workspace/Output/Raise/Raise_original_151a2cf8-8c25-4218-a3d6-6a563ce02344.JPG	Raise	Raise	100.0%	Gany	0.0%	Nothing	0.0%

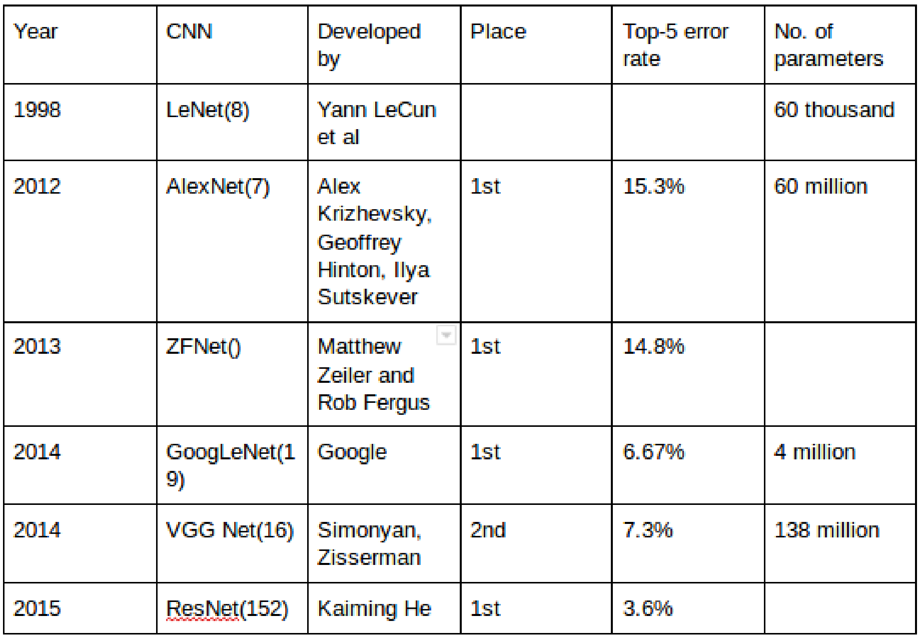
Discussion

The original images are almost covering full dog body, using augment code can easily generate different angle and different part of dog body images. There is no problem to use rotate, flip and resize functions to generate new images, but crop image function can cause some image problems if the new image didn't include the target object at all. The image source quality is very important for Deep Learning training result, the manually checking was applying all these generated images to make sure no any nothing image mixed in dog image classes.

To achieve the best result, the LeNet network was not used because only 28x28 image size and gray color can be used. But AlexNet and GoogLeNet, both networks support 256x256 color image.

Both AlexNet and GoogLeNet have been tested for classification of dog image, GoogLeNet showed better accuracy than AlexNet at epoch = 20. According to the research paper from Siddharth Das[4], GoogLeNet had a better performance than AlexNet.

CNNs Architectures:LeNet, AlexNet, VGG, GoogLeNet, ResNet comparing table:



Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

So GoogLeNet was selected to continue training and testing. Set epoch at 5, 20, 30 and 50 to train the GoogLeNet network, the best result is at epoch = 50, Batch size = 50.

Changed batch size to 50, can reduce training time and use less memory.

Conclusion

Using augmentation code to generate image is a good and fast way to get a large number of image.

The result used GoogLeNet model (with epoch = 50, Batch size = 50 and test dataset = 1% of total images) and plus image augmented images are very satisfied in classification dog images (All three classes accuracy are 100%).

Future Work

1. Install Nvidia DIGITS system on local PC instead of using cloud resource, so there is no time limitation to implement and test different projects and models.
2. Include testing object detection and segmentation implementation, and deploying the model on Jetson TX2 board and testing them in real world environment.

References

[1] S. Perkins, A. Walker and E. Wolfart, Classification "https://homepages.inf.ed.ac.uk/rbf/HIPR2/classify.htm" 2003

[2] Nvidia, DIGITS workflow "https://developer.nvidia.com/digits" 2018

[3] Marcus D. Bloice, Augmentor "https://github.com/mdbloice/Augmentor" 2018

[4] Siddharth Das, CNNs Architectures:LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ... "https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5" 2017