

Robotics Software Engineer Nanodegree Term 2 - Project 1: Robotics Inference

Mithi Sevilla - medium.com/@mithi

ABSTRACT

I have used Nvidia DIGITS 6.0 on an AWS AMI to produce models that are capable of detecting and classifying objects in images. I have used the supplied dataset to train an existing model to classify if an image is a candy box, a bottle, or nothing. I have also collected images of myself, other people that are not me, and image images without people (only background) to train a model to classify appropriately. Upon evaluation, the model for the supplied data meets the required accuracy of above 75% (around 78%), and required inference time of below 10 ms (around 6 ms) on p2.xlarge instance. The model for my own data set seems to just be a notch above guessing.

INTRODUCTION

DIGITS (the Deep Learning GPU Training System) is a web application for training deep learning models. It utilizes transfer learning from existing pretrained models such as LeNet, AlexNet, and GoogLeNet. The idea of transfer learning is that it is more efficient to use a model that has learned to recognize cars as a starting point to train it to recognize trucks than start from scratch.

Recognizing objects is an important first step in the field of robotics because a good understanding of its real-time environment helps the robot make better decisions very much like a human being. More notably, it is not only important that a robot recognizes objects correctly, but also that it does this quickly and in a timely manner since the environment changes and robots are expected to respond in real time. We should pay attention not only on the model's accuracy but also its inference time given the limited computing power of a robot's brain.

If I were to build a home robot, the people who live in that home will be interacting with that robot all the time. I think it would be very useful for robot to recognize and differentiate the people that it will always be interacting with. This is why I chose to recognize myself. I also wanted to recognize my housemates but for lack of time, maybe in the future.

BACKGROUND

As a starting point, we can choose between three pretrained networks - LeNet (1998), AlexNet (2012), and GoogleNet (2014). The intended images for Lenet is grayscale with a resolution of 28x28 while the other two expect 256x256 colored images. I chose GoogleNet mainly because it is the most recent, and I know Google is a large company heavily invested in this - with a lot of training data and computing resources. LeNet is insufficient because my pictures are colored by default and also it is trained to only recognize simpler things like handwritten numbers. Between AlexNet and GoogleNet, it is the latter which won ILSVRC - a large scale visual recognition challenge - more recently.

To train an existing network further, there are a lot of parameters you can choose from. Most important ones, I believe, are the epoch, learning rate, and solver type.

An epoch is one complete pass of all the data in our data set. So 30 epochs means we pass the dataset 30 times. We must be aware of this since a large epoch size might overfit our network while a smaller size will underfit it. Overfit means that it might work very well at our training data set but won't generalize to new data, while underfit means that the model works poorly both with our training data and new data.

The intuition behind learning rate is how quickly a network abandons old beliefs for new ones. We want to find a learning rate that is low enough that the network quickly becomes useful, but high enough that you don't have to spend years training it. If the learning rate is too high then it becomes like a wishy-washy person who always misses the point every time you explain things differently or provide a new example of an idea. If the learning rate is too low then it's like a stupid person who can eventually understand something but takes so long it's annoying. We want the network to have a learning rate (or sometimes an adaptive learning rate) such that it behaves like a reasonably intelligent person.

Solver type or optimizer type like SGD, Adam, Nesterov Momentum is the algorithm used to update the weights of the network based on the training data so that the final model converges into something useful fast. Each has its pros and cons, so usually I just try each of them and see which gets me the best performing model.

DATA ACQUISITION

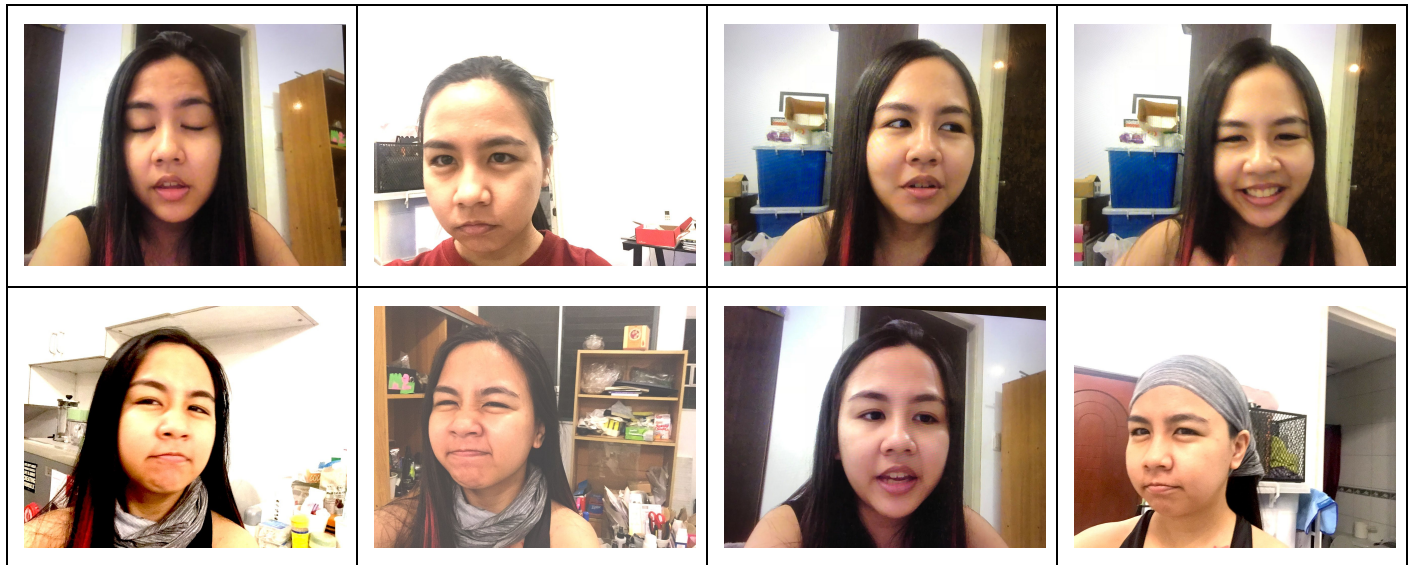
So I used my phone to get low resolution images of my face. To ensure the quality of my data I made sure I had the following covered:

- Pictures of me at a few reasonable portrait angles (front, few angles between $\frac{3}{4}$ and front from left and right size, looking up and looking down) .
- Pictures of me at various background around the house,
- Pictures of me with my hair down, in a ponytail, and with a bandana so it will not recognize me based on my current hairstyle
- Pictures of me with some variations in lighting (warm light, white light etc)
- Pictures of me with some of my common facial expressions (smiling)
- Picture of me with wet hair and dry hair!
- Pictures of me wearing three types of shirt tops so it will not recognize me based on my shirt.
- Some pictures with my mouth open and my mouth closed, with eyes open and closed
- Still frames of some videos of myself talking/teaching

I took some photos around the house without any people. I also got pictures of other of my friends off facebook and google image search. I did this to ensure that my face gets differentiated from other people and not get recognized as a generic face.

The important characteristic is to consider what could be patterns in my pictures that could be picked up by the model which has nothing to do with me. It is important that the pictures captures what I think are defining characteristics of my face which is my eyes, nose, cheeks and mouth and NOT my hair, clothing or skin color. I selected 197 images of myself, 164 images of background, and 125 images of other people's faces.

Examples: My Face



Examples: Other people's faces and background



RESULTS

Supplied Data Results

With GoogLnet as a starting point, here is the result and graph of training of around 7.4 epochs (*the training got aborted inadvertently*) and a learning rate of 0.001 with default parameters for everything else such as SGD as solver type.

```
[ubuntu@ip-172-31-25-28:~$ cd digits
[ubuntu@ip-172-31-25-28:~/digits$ evaluate
```

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

[Please enter the Job ID: 20171129-141915-ec4f

Calculating average inference time over 10 samples...

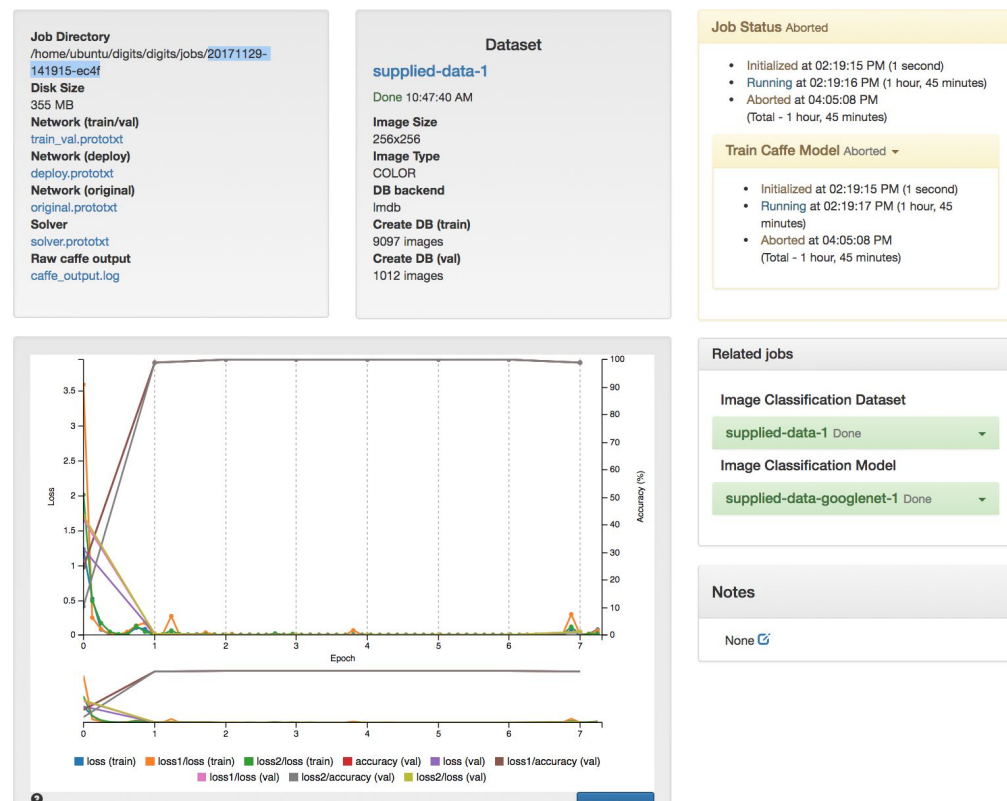
```
deploy: /home/ubuntu/digits/digits/jobs/20171129-141915-ec4f/deploy.prototxt
model: /home/ubuntu/digits/digits/jobs/20171129-141915-ec4f/snapshot_iter_1995.caffemodel
output: softmax
iterations: 1
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
```

```
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.55904 ms.
```


Calculating model accuracy...

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current				
			Dload Upload	Total	Spent	Left	Speed				
100	11126	100	9305	100	1821	155	30	0:01:00	0:00:59	0:00:01	1961


Your model accuracy is 78.2608695652 %




Test Results for Sample Images of Bottle, Candy Box, and Nothing (*Included in the Training Set*)




Predictions	
Bottle	99.91%
Candy Box	0.09%
Nothing	0.01%




Predictions	
Bottle	99.95%
Candy Box	0.05%
Nothing	0.0%




Predictions	
Bottle	99.99%
Candy Box	0.01%
Nothing	0.0%




Predictions	
Bottle	99.97%
Candy Box	0.03%
Nothing	0.0%




Predictions	
Bottle	99.99%
Candy Box	0.01%
Nothing	0.0%



Predictions	
Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%



Predictions	
Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%



Predictions	
Nothing	99.99%
Candy Box	0.01%
Bottle	0.0%

Here are test images from the internet that I used which the model got right



Predictions	
Bottle	98.28%
Candy Box	1.72%
Nothing	0.0%

1



Predictions	
Bottle	98.72%
Candy Box	1.28%
Nothing	0.0%

2



Predictions	
Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

3




Predictions	
Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

4



Predictions	
Nothing	99.6%
Candy Box	0.4%
Bottle	0.0%


5



Predictions	
Nothing	100.0%
Candy Box	0.0%
Bottle	0.0%


6

Here are test images from the internet that I used which the model got WRONG




Predictions	
Bottle	100.0%
Candy Box	0.0%
Nothing	0.0%

1



Predictions	
Bottle	85.43%
Candy Box	12.07%
Nothing	2.5%

2



Predictions	
Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

3

A box of Odwalla fruit juice, specifically the blueberry flavor, standing next to other Odwalla boxes in various flavors like orange and pink.

Predictions

Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

A bottle of wine, specifically a bottle of Pinot Noir, standing on a surface.

Predictions

Candy Box	87.28%
Nothing	11.97%
Bottle	0.77%

A bottle of Reese's Peanut Butter, specifically the Reese's Peanut Butter variety, standing on a surface.

Predictions

Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

A sign that says "NOTHING" in a desert, standing on a sandy surface.

Predictions

Candy Box	100.0%
Nothing	0.0%
Bottle	0.0%

A dark, abstract painting, possibly a landscape or a scene with figures, standing on a surface.

Predictions

Bottle	99.86%
Candy Box	0.14%
Nothing	0.0%

My Own Data Results

Adaptive Gradient, 30 epochs, Adaptive Starting Learning Rate of 0.01 with GoogLeNet as a base

my-data-model-adagrad-1

Owner: ubuntu

Clone Job Delete Job

Job Directory

/home/ubuntu/digits/digits/jobs/20171130-150146-8b9f

Disk Size

1.19 GB

Network (train/val)

[train_val.prototxt](#)

Network (deploy)

[deploy.prototxt](#)

Network (original)

[original.prototxt](#)

Solver

[solver.prototxt](#)

Raw caffe output

[caffe_output.log](#)

Dataset

my-data-1

Done 02:57:22 PM

Image Size

256x256

Image Type

COLOR

DB backend

lmdb

Create DB (train)

411 images

Create DB (val)

72 images

Job Status Done

- Initialized at 03:01:46 PM (1 second)
- Running at 03:01:47 PM (3 minutes, 40 seconds)
- Done at 03:05:28 PM

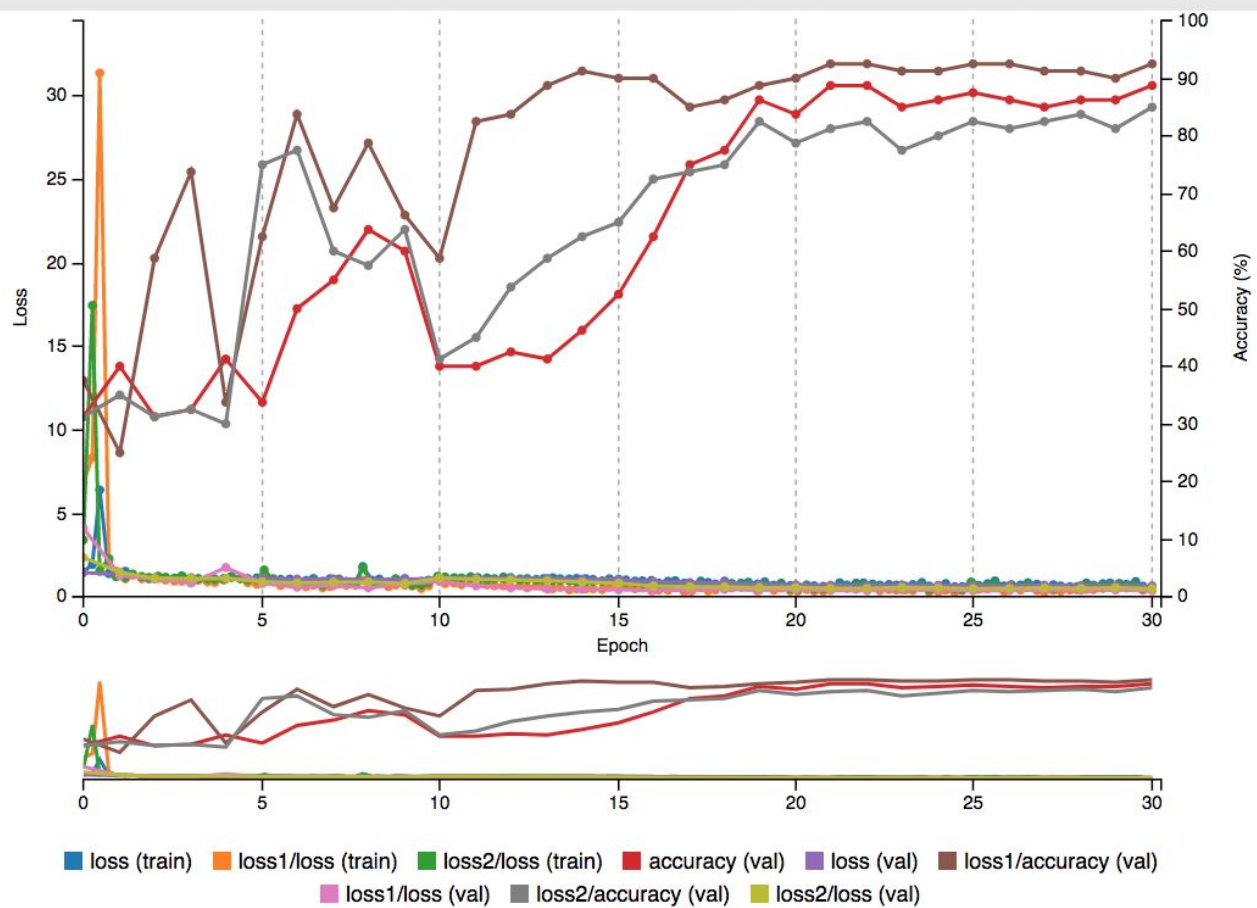
(Total - 3 minutes, 42 seconds)

Train Caffe Model Done

Related jobs

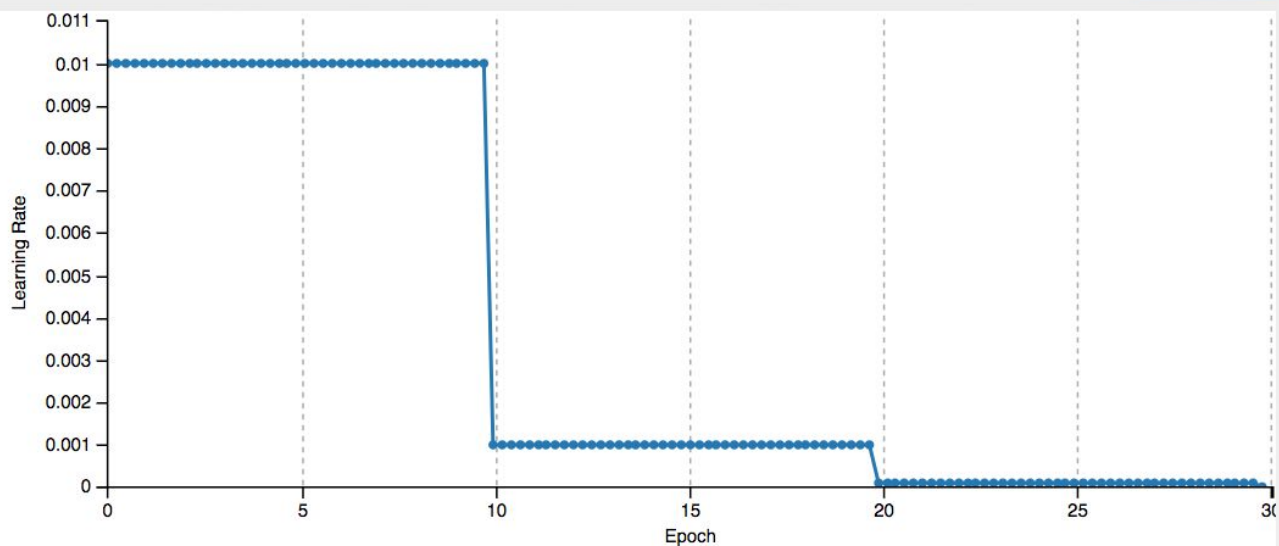
Image Classification Dataset

my-data-1 Done



?

[View Large](#)



my-data-model-adagrad-1 Image Classification Model



Predictions

my face	55.13%
other people face	25.53%
background	19.33%

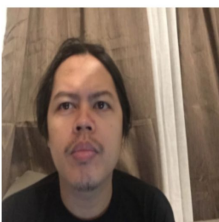
my-data-model-adagrad-1 Image Classification Model



Predictions

my face	48.57%
other people face	32.25%
background	19.17%

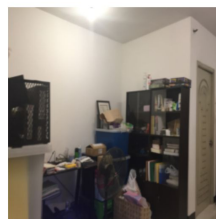
my-data-model-adagrad-1 Image Classification Model



Predictions

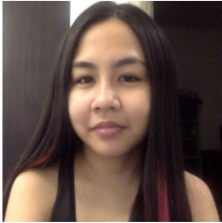
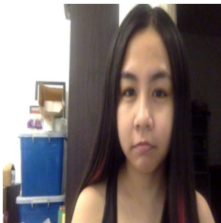

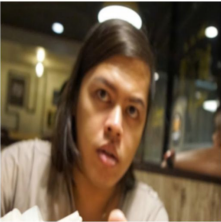
other people face	51.36%
my face	32.93%
background	15.71%

my-data-model-adagrad-1 Image Classification Model



Predictions

background	87.34%
my face	6.71%
other people face	5.95%

my-data-model-adagrad-1 Image Classification Model  <table> <tr><th colspan="2">Predictions</th></tr> <tr> <td>other people face</td><td>82.01%</td></tr> <tr> <td>my face</td><td>24.54%</td></tr> <tr> <td>background</td><td>13.46%</td></tr> </table>	Predictions		other people face	82.01%	my face	24.54%	background	13.46%	my-data-model-adagrad-1 Image Classification Model  <table> <tr><th colspan="2">Predictions</th></tr> <tr> <td>my face</td><td>38.1%</td></tr> <tr> <td>other people face</td><td>33.19%</td></tr> <tr> <td>background</td><td>28.71%</td></tr> </table>	Predictions		my face	38.1%	other people face	33.19%	background	28.71%
Predictions																	
other people face	82.01%																
my face	24.54%																
background	13.46%																
Predictions																	
my face	38.1%																
other people face	33.19%																
background	28.71%																
 <table> <tr><th colspan="2">Predictions</th></tr> <tr> <td>my face</td><td>55.06%</td></tr> <tr> <td>other people face</td><td>25.89%</td></tr> <tr> <td>background</td><td>19.04%</td></tr> </table>	Predictions		my face	55.06%	other people face	25.89%	background	19.04%	 <table> <tr><th colspan="2">Predictions</th></tr> <tr> <td>my face</td><td>62.87%</td></tr> <tr> <td>other people face</td><td>28.09%</td></tr> <tr> <td>background</td><td>19.04%</td></tr> </table>	Predictions		my face	62.87%	other people face	28.09%	background	19.04%
Predictions																	
my face	55.06%																
other people face	25.89%																
background	19.04%																
Predictions																	
my face	62.87%																
other people face	28.09%																
background	19.04%																

DISCUSSION

Supplied Data Discussion

For the supplied data set, at first I chose an epoch of 30, and the default learning rate of 0.01 using a solver type of a SGD as these were the defaults. I did not get good results (all NAN). I then used a smaller learning rate of 0.001 and then the job got aborted at around epoch 7.4 . Because i didn't want to retrain again and based on the graph it looks like it learned pretty well, I decided to test and evaluate this current model. Turns out it works pretty well even though imperfect; it still has room for a lot of improvement.

I noticed that the supplied data set has bottles of only three different kinds taken at various angles - a bottle of Odwalla, a specific coconut juice tetrapack, and a certain dressing brand. For the candy box, it's just Reese and Whoppers. For nothing, it's just white with a bit of shadow.

I tested various images from the internet while making an educated guess beforehand if they will be predicted correctly or not. I was pleasantly surprised that it had correctly predicted the two skittles boxes even though skittles wasn't in the dataset. I guess the model correlated the rectangular prism shape and bright colors with candy boxes. This is also my theory as to why it got the Odwalla set of bottles (item 4) wrong. It also correctly predicted a different tetrapack coconut juice drink correct which has a similar shape but not very different branding style so (item 2) and a similarly a different type of dressing with but similar container shape.

Also, I used a logo of Odwalla which is not a bottle, I correctly predicted that it would get this wrong as the model associated this logo to bottle. It predicted this with great confidence as well (100%). When I used a bottle which is similar shape as a dressing bottle but with a logo of Reese, it was 100% sure that that it was a candy box and not a bottle. For wrong items 2 and 8 they were abstract paintings but had high confidence that it was a bottle for some reason. I understand a bit why item 2 looks like a bottle but a bit bewildered about item 8. Item 3 and 5 were bottles but are mistaken as candy boxes with high confidence. Maybe it's because of the colorfulness of the Pepsi bottle. A dark sign with "nothing" is mistaken as a candy box perhaps because it is shaped as a rectangular prism. All in all this means that the model is not that good and should still be trained with more data to perform better.

My Own Data Discussion

As I've mentioned earlier, I took images of myself, and 164 images of background using my phone. I also took 125 images of other people's faces from facebook and google image search. I removed similar looking images which I took with my phone with burst shot which reduced the total number of images of myself to 197.

As you might have noticed, the model is behaving very poorly. Cropping the image makes the model perform worse which makes me realize it might be correlating my face to some of the backgrounds of the images. The culprit here is that not enough data was used for training for the model to generalize to new pictures. Also, I should have

cropped the images of faces so that it is featured more prominently than the background. Data augmentation techniques, needless to say, should also be used.

What is good is that it identifies that a face is in the picture with a high confidence based on the sample tests above. However the model has a difficult time distinguishing my face from everyone else.

For this project, inference time is almost as important as accuracy because of the user experience aspect. If a robot takes a long time to identify the face of a person, the person will be impatient and will not have a good time interacting with the robot.

FUTURE WORK

It is painfully obvious that the two models are performing subpar. At its current state, this is not a commercially viable product because it is not accurate enough. There are many things that should be done to improve the accuracy of the model. This includes but not limited to:

- gather more training images
- closely inspect the data, and carefully check if unnecessary patterns of the images exist which would readily confuse the model.
- apply data augmentation techniques - <http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf> (*The Effectiveness of Data Augmentation in Image Classification using Deep Learning*)

As I've said earlier, for this project inference time is almost as important as accuracy because of the user experience aspect. If a robot takes a long time to identify the face of a person, the person will be impatient and will not have a good time interacting with the robot. On the other hand, if it just keeps failing at classifying the face incorrectly, then what's the point? We have to balance the tradeoff between inference time and accuracy. Instead of aiming best accuracy-slow inference time or worst accuracy - fast inference time, we should aim for good accuracy - good inference time.

Having a robust model that recognizes a specific face can be very useful for security purposes and automated identification. When a robot can recognize you, it can tailor its interaction with you specific to your personality the way human beings do.

"If a child sees 10 examples of cats and all of them have orange fur, it will think that cats have orange fur and will look for orange fur when trying to identify a cat. Now it sees a black a cat and her parents tell her it's a cat (supervised learning). , it could be the case that the child will think that this black cat is an outlier and cats are still orange." - <https://www.quora.com/What-is-the-learning-rate-in-neural-networks>

REFERENCES

Guiding Doc (Restricted Access)

https://docs.google.com/spreadsheets/d/1NGMuV_ReCH1TIQXcoxT5kaQo2GTGsbTTpwT30TFNksE/edit?ts=5a1d7f1a#gid=0

Beta Tester Info (Restricted Access)

https://docs.google.com/document/d/1PdWN-GBD8n8jGsg20S_wtB67DrDKeUBZL5_-fEX3S4/edit?ts=5a1d7f1b

NVIDIA DIGITS Training Guide

<https://github.com/dusty-nv/jetson-inference/blob/master/README.md#importing-classification-dataset-into-digits>

<https://github.com/S4WRXTTCS/jetson-inference>

<https://github.com/NVIDIA/DIGITS>