Robotics Inference Project

Gene Foxwell; Udacity Robotics Software Engineering Inference Project.

ABSTRACT

Recognizing meta information about the environment could have many applications for SLAM and Augmented Reality based systems. This project builds an inference engine intended to extract the category of "room" an autonomous robot finds itself in based on input from a standard web camera. Data was collected with a built-in camera on a laptop. Three models where trained using the NVIDIA DIGITS application and then tested against a set of images collected on a simulated drive through of the environment. Results where then evaluated based on how often the systems best prediction matched the ground truth of the room it represented.

INTRODUCTION

The ability to recognize the context of devices surroundings is a useful tool to have for many applications. Consider for example, a home robot that knows when its "in a Livingroom" and can use that information to determine what actions are possible, or convey its location in a meaningful way. (Rather than I am at position x,y,z on the map, it could simply relay to the user "I am in the Living Room"). Such a feature would also be useful in AR applications where the system could adjust the data it is showing the user based on the room or area it is in.

This project addresses the problem of context inside a house by making use of an inference system built with the help of NVIDIA's DIGITS software. Four models where built using the software, one model built on a generic test set provided by Udacity, and three built on the data collected by the experimenter. These models where then evaluated using the tools available in the NVIDIA DIGITS interface and the Udacity student workspace. Once the models where suitably trained a simulated run through of the house was done on the Jetson TX2 hardware to evaluate how effective a real-world robot might have been at determining its context based on the input from its Camera alone.

At least one other notable approach to solving this problem at found described in the book Semantic Labeling of Places with Mobile Robotics[1] (Oscar Martinez Mozos). Martinez's work focused primarily on labeling maps generated using a laser range finder via various machine learning methods. Although camera input is briefly considered to augment the data coming in from the range scanners it is not the primary sensor being considered. The difference in this work is that it is attempting to provide the semantic label using camera input alone, without any prior knowledge of the robots pose, geometric information, or past trajectory.

As this is intended as a system for providing meta-data or other contextual information to an already running system high speed performance was not a consideration. Any system using this model could

[1] Oscar Martinez Mozos, https://www.researchgate.net/publication/29755036_Semantic_Labeling_of_Places_with_Mobile_Robots

simply be updated periodically to make sure that it was still using the correct context or assigning the correct label to the current section of the map it was in. With that in mind however, if the system is too slow it may cause the end user to become frustrated or lose interest and so cannot be allowed to be infinitely slow. Based on common UX guidelines, the goal of this inference system was to provide meta data at a minimum rate of one classification per second[2].

BACKGROUND

This project was split into two distinct stages. In stage one the NVIDIA DIGITS system was experimented with using a data set provided by the Udacity classroom to build a classifier capable of discriminating between three classes at roughly 75% accuracy with a latency of no more than 10ms. In stage two the NVIDIA DIGITS system was leveraged to build the inference model for discriminating between the different rooms in the house. Once built this model was deployed to a Jetson TX2 to evaluate its performance on a simulated dataset.

STAGE 1 – UDACITY PROVIDED DATASET

Udacity provided a conveyor belt dataset for use in familiarizing oneself with the DIGITS workflow. This dataset was meant to represent different items a sorting robot might find on a conveyer belt. These items where representing by approximately 4500 hundred color images of size 256 x 256 pixels split into three classes: Bottle, Candybox, and Nothing.

To hit the target values of 75% (or better) accuracy with a 10ms (or less) response time, the dataset was loaded into the DIGITS framework leaving approximately 20% of the data aside for validation purposes. Once loaded DIGITS was used to train a classification model against this data. For this classification problem, the following parameters where set:

Model: GoogLeNet[3]

Epochs: 10

Learning Rate: 0.01

Step: Exponential Decay with a gamma of 0.99

[2] https://www.nngroup.com/articles/response-times-3-important-limits/ - Longer than one second and the user may lose interest, or broken from their flow state causing frustration and a feeling of a sluggish application.
[3] Going Deeper with Convolutions; Christian Szegedy et all; https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf

These parameters where chosen by trial and error. A few initial attempts using both the built in AlexNet[4] and GoogLeNet using other styles of learning rates where attempted, but these tended to have disappointing results. A learning rate of 0.01 using the Step-Down method for example, only achieved a maximum accuracy of roughly 64% even after 30 iterations, while with an Exponential Decay strategy both GoogLeNet and AlexNet where able to achieve passing results. However, for AlexNet far more training time was needed (roughly 30 iterations rather than 10). As GoogLeNet appeared to take less training time for similar results it was used as the chosen solution for this problem.

Once training was completed, the Udacity environment was used to evaluate the results using the provided evaluate program. Evaluation showed that the trained model was able to achieve an accuracy of 75.4% and a response time averaging between 5ms and 6ms. A watermarked summary of this can be seen in the results section. In addition, both the standalone watermarked image and the downloaded model from the DIGITS server have been included with files that accompany this project.

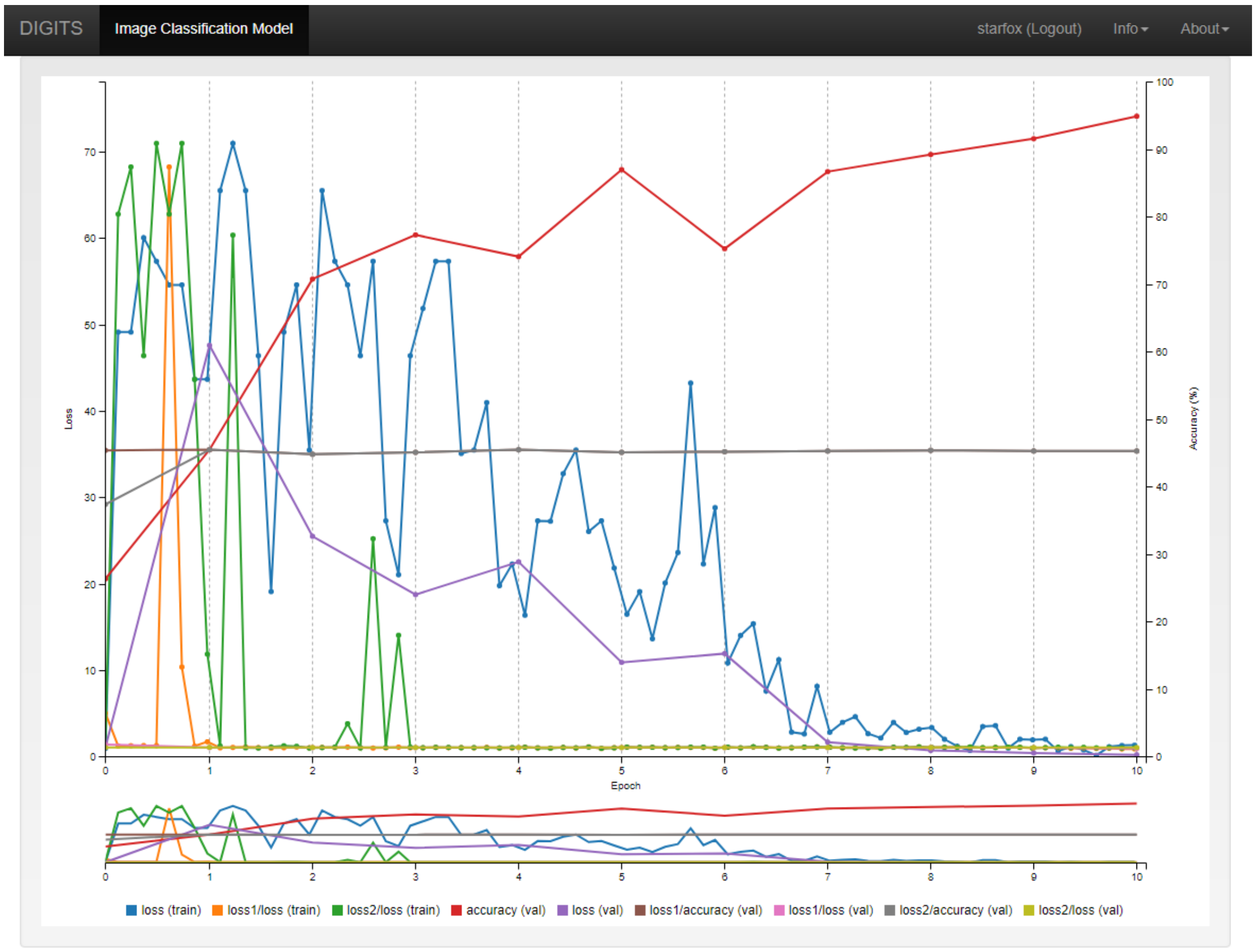


Figure 1 Loss from 10 Epoch Trian of GOOGLENET

[4] ImageNet Classification with Deep Convolutional Neural Networks; Alex Krizhevsky et all; https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Stage 2: Identifying Rooms

This stage of the project focused on building an inference system for identifying the room of a house that an image belongs to. For practical reasons this was limited to be a single home (the experimenters) to which the experimenter could reasonably expect access to. As outlined in the Data Acquisition section of this paper, approximately 3500 images where collected of the houses various rooms, and divided into five distinct categories:

1. Living Room
2. Dining Room
3. Kitchen
4. Hallway
5. Bathroom

These where chosen to represent a reasonable subset of the house that a home service robot, or AR device might commonly be used in. (It is questionable whether such devices would be welcome in user's bedrooms given that they rely on cameras to operate). This data was suitably labelled and uploaded to the Udacity DIGITS workspace and converted into a DIGITS dataset. 15% of the data in this dataset was reserved for the validation set while a further 10% was reserved to be part of the test set[5].

Once created, several models where experimented with, each using a learning rate of 0.01 combined with exponential decay. This choice of learning rate was made based on the results of training a model on the Udacity dataset in the previous stage. Each model was trained until it appeared to converge. The corresponding loss curves for the training of these models is included in the Results section.

With the models built, the next step was to evaluate how a real autonomous system equipped with the Jetson TX2 might perform using this model. While a Jetson TX2 was available, a practical way of moving it around the experimenter's house was not available. To get around this limitation the experimenters laptop was used to collect data for a simulated tour of the household. These where classified by hand and labeled in order from the start of the tour to the end of the tour. This data was then uploaded to the Jetson TX2 via FTP from a host computer.

The actual simulation was done by using a modified version of the imagenet-console code provided in the jetson-inference git repository called imagenet-batch. As the name suggests the imagnet-batch program processes a set of images as a "batch" and outputs its best guess for each image. Rather than outputting a collection of images with their predictions super-imposed (as imagenet-console does) the imagenet-batch program simply outputs a list of filename / prediction pairs. This makes assessing the results of the batch significantly easier as one does not need to load each individual image to see what was output. Code for the imagenet-batch program has been included with the supporting files of this project.

[5] This was used to generate the confusion matrices in the results. The suitability of using DIGITS test set for this use was suggested by https://groups.google.com/forum/#!topic/digits-users/4xhI5w8bL-E

A special case occurred when the VGG-16 based model was used on the system. It appears that the trained model requires 1000 classes to be defined, where for obvious reasons this project only defined 5. A work around was made in which the imageNet.cpp file was edited to have this restriction removed (line 152 of imageNet.cpp was commented out and the jetson-inference project was rebuilt). Once this additional step was taken, testing for the VGG-16 model was able to continue as normal.

Output from the imagenet-batch program was then parsed using an additional python script (included). This script simply compared the predicted class from the imagenet-batch to the ground truth assigned by the experimenter (this ground truth was encoded in the script). Failed matches were kept track of and printed to the stdout and on completion the script an accuracy value was provided.

DATA ACQUISITION

Data acquisition was done by making use of the built-in webcam on an older HP Laptop running kunbutu. To acquire data a C++ program was written to make use of the OpenCV framework. Once started the program randomly sampled images from the camera feed, scaled them to 256x256 pixels (from an original size of 640x480 pixels) then saved them to disk. While this program was running, the experimenter held the laptop roughly waist height and walked the laptop around the various rooms of interest to simulate the different points of view that a robot might have. Once completed the program was shut down and the images sorted manually into their respective categories. Approximately 3500 images where collected using this method.

Occasionally during the process of data collection, the experimenter’s face / body would obscure the cameras view (typically while he was activating or deactivating the data collection program on the laptop). These images where removed from the final dataset. Some images also contained the experimenters dog, which after some thought were left in as they did not block any key features regarding the room and any autonomous vehicle existing in the home would almost certainly need to contend with the owner’s pets.

With respect to the class distribution, as it can clearly be seen from figure 1 there is some imbalance between the classes. Specifically, the “BATHROOM” class is noticeable smaller than its counterparts. This is caused by the geometry of the house used to collect the data – some rooms are simply too small, or differently arranged to allow for a large variety of viewing angles to be collected.

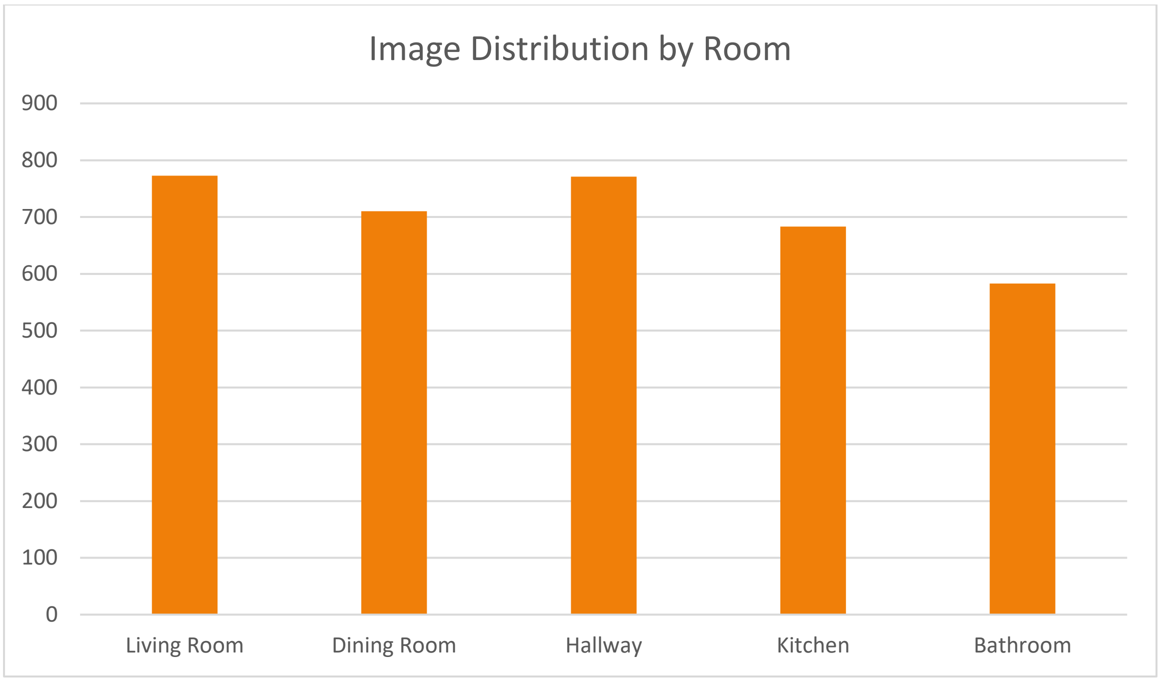
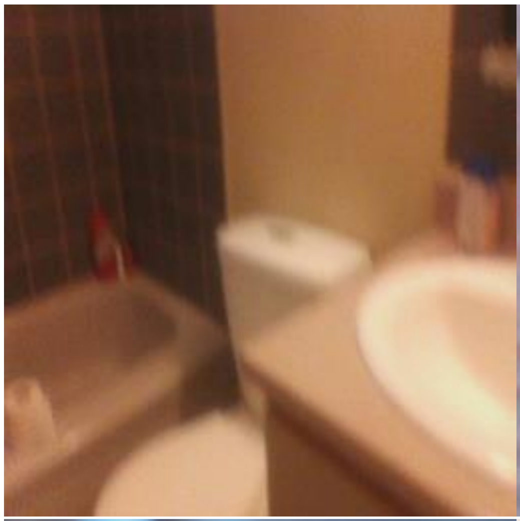
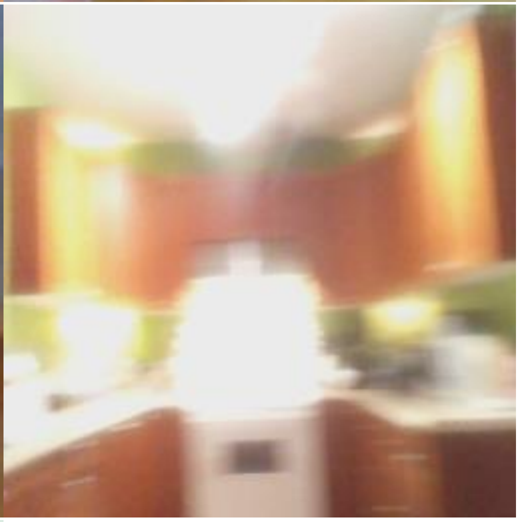


Figure 2

As the currently available Jetson TX2 was not yet converted into mobile form, the model was tested on a simulated walkthrough of the household. Data for this simulation was collected by running the data collector routine on the experimenter's laptop and simply doing a walkthrough of the house. 341 images where collected for this purpose. These images where not used during training.

Sample images from each of the five categories are attached below.

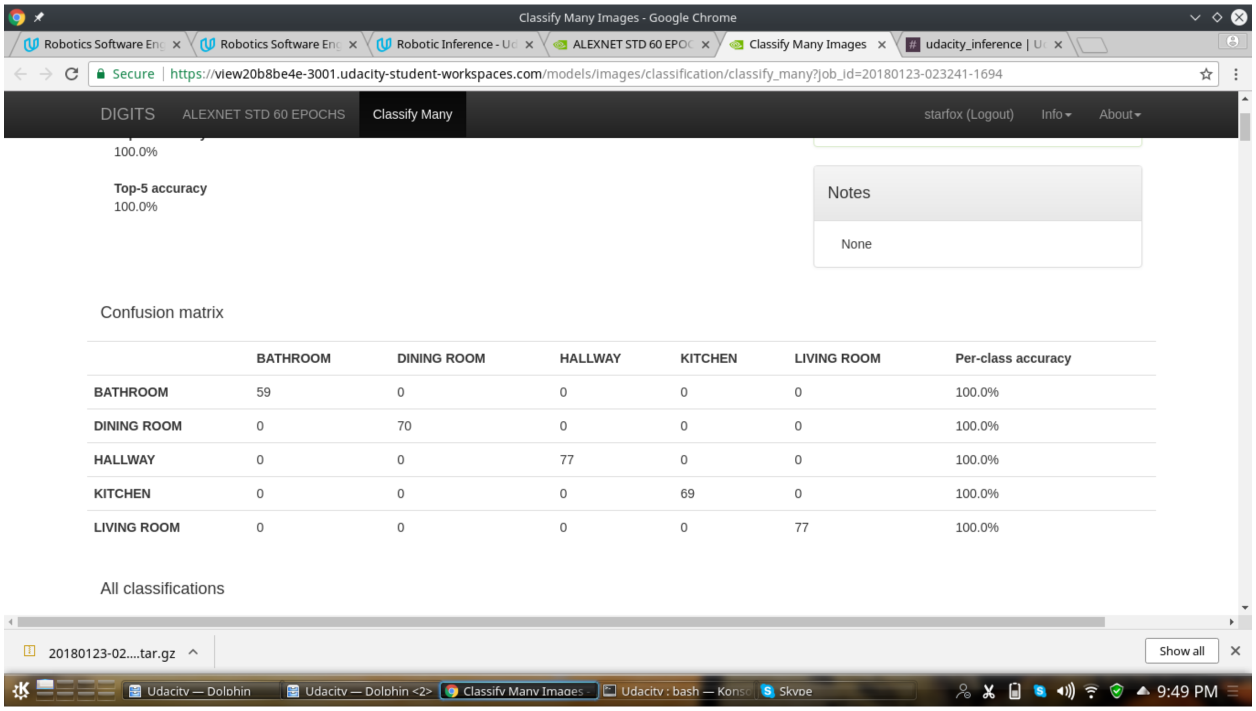




RESULTS

Results of evaluating the trained GoogLeNet model on the Udacity provided dataset have been included in figure 7. As can be seen in the image; using the provided evaluate function to test the model against a dataset not seen before by the model the GoogLeNet model was able to obtain 75.4% accuracy with a response time between 5ms and 6ms. As these were well within the target metrics for this task no further experimentation was done for this task.

Models generated for room inference in the DIGITS interface where tested in two stages. First, they were tested using the classify many option in DIGITS using the train.txt as the model list. The confusion matrices that this generated have been included below.


Classify Many Images - Google Chrome
https://view20b8be4e-3001.udacity-student-workspaces.com/models/images/classification/classify_many?job_id=20180123-023241-1694
DIGITS
ALEXNET STD 60 EPOCHS
Classify Many
starfox (Logout)
Info
About
Notes
None
20180123-02....tar.gz
Show all
9:49 PM

100.0%

Top-5 accuracy
100.0%

Confusion matrix

	BATHROOM	DINING ROOM	HALLWAY	KITCHEN	LIVING ROOM	Per-class accuracy
BATHROOM	59	0	0	0	0	100.0%
DINING ROOM	0	70	0	0	0	100.0%
HALLWAY	0	0	77	0	0	100.0%
KITCHEN	0	0	0	69	0	100.0%
LIVING ROOM	0	0	0	0	77	100.0%

All classifications

Figure 3 AlexNet Model Confusion Matrix

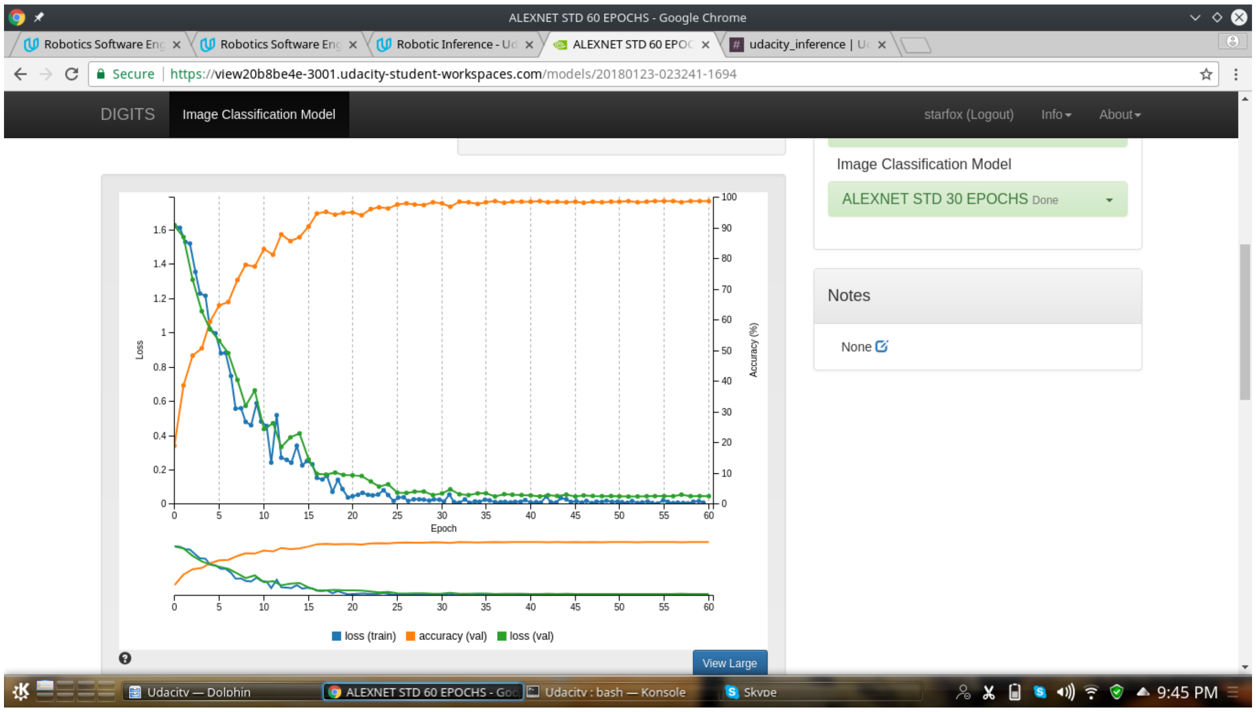
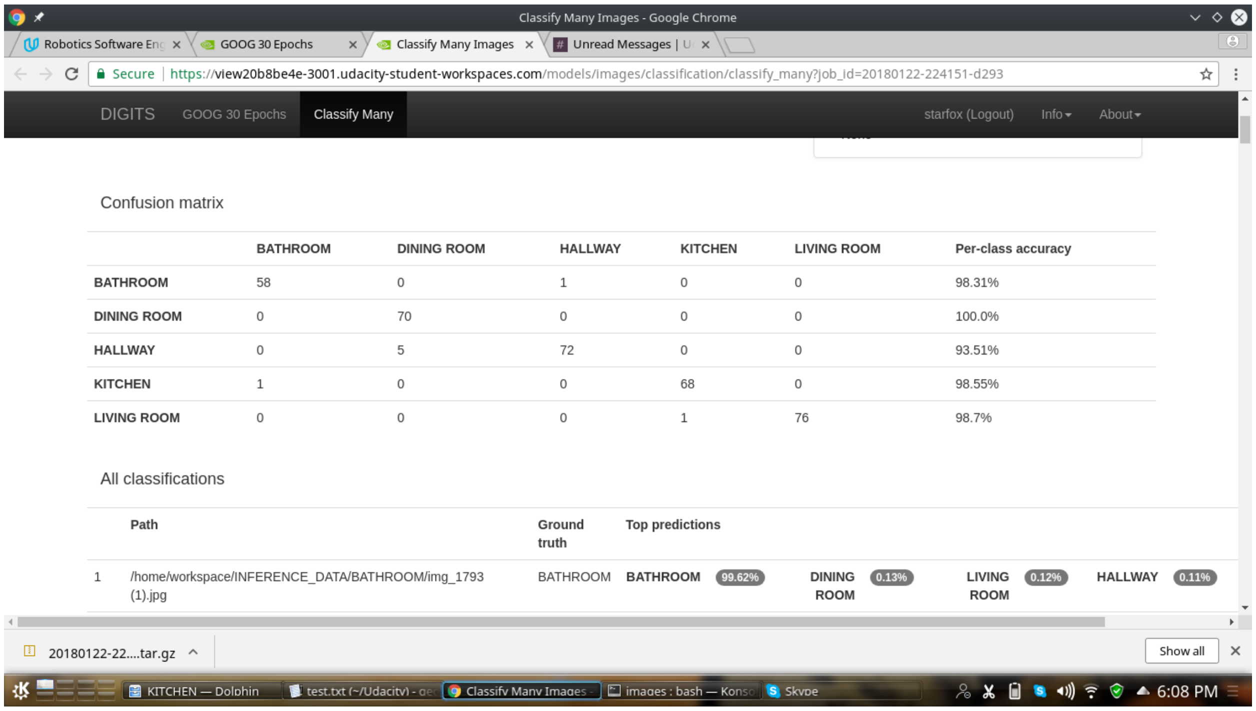


Figure 4 AlexNet Model Loss curves


Classify Many Images - Google Chrome
https://view20b8be4e-3001.udacity-student-workspaces.com/models/images/classification/classify_many?job_id=20180122-224151-d293
DIGITS
GOOG 30 Epochs
Classify Many
starfox (Logout)
Info
About
20180122-22....tar.gz
Show all
6:08 PM

Confusion matrix

	BATHROOM	DINING ROOM	HALLWAY	KITCHEN	LIVING ROOM	Per-class accuracy
BATHROOM	58	0	1	0	0	98.31%
DINING ROOM	0	70	0	0	0	100.0%
HALLWAY	0	5	72	0	0	93.51%
KITCHEN	1	0	0	68	0	98.55%
LIVING ROOM	0	0	0	1	76	98.7%

All classifications

	Path	Ground truth	Top predictions							
1	/home/workspace/INFERENCE_DATA/BATHROOM/img_1793 (1).jpg	BATHROOM	**BATHROOM**	99.62%	**DINING ROOM**	0.13%	**LIVING ROOM**	0.12%	**HALLWAY**	0.11%

Figure 5 GoogLeNet Model Confusion Matrix

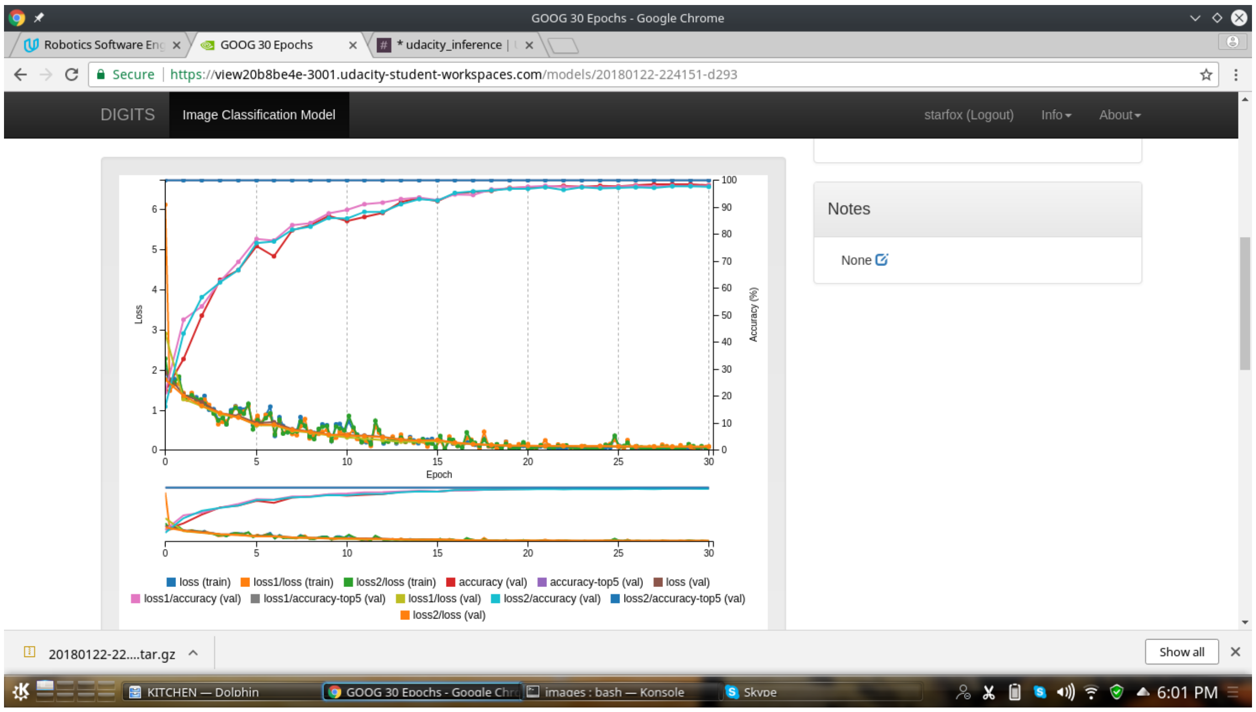
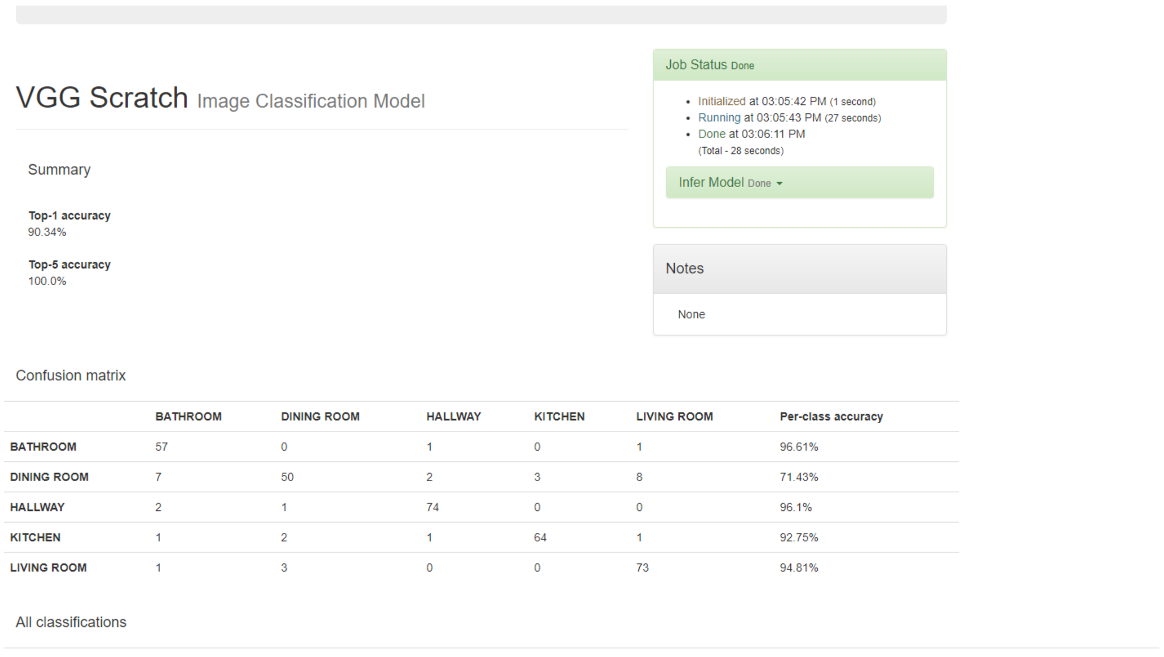


Figure 6 GoogLeNet Loss curves



VGG Scratch Image Classification Model

Job Status Done

- Initialized at 03:05:42 PM (1 second)
- Running at 03:05:43 PM (27 seconds)
- Done at 03:06:11 PM
  (Total - 28 seconds)

Infer Model Done

Summary

Top-1 accuracy
90.34%

Top-5 accuracy
100.0%

Notes

None

Confusion matrix

	BATHROOM	DINING ROOM	HALLWAY	KITCHEN	LIVING ROOM	Per-class accuracy
BATHROOM	57	0	1	0	1	96.61%
DINING ROOM	7	50	2	3	8	71.43%
HALLWAY	2	1	74	0	0	96.1%
KITCHEN	1	2	1	64	1	92.75%
LIVING ROOM	1	3	0	0	73	94.81%

All classifications

Figure 7 VGG Confusion Matrix

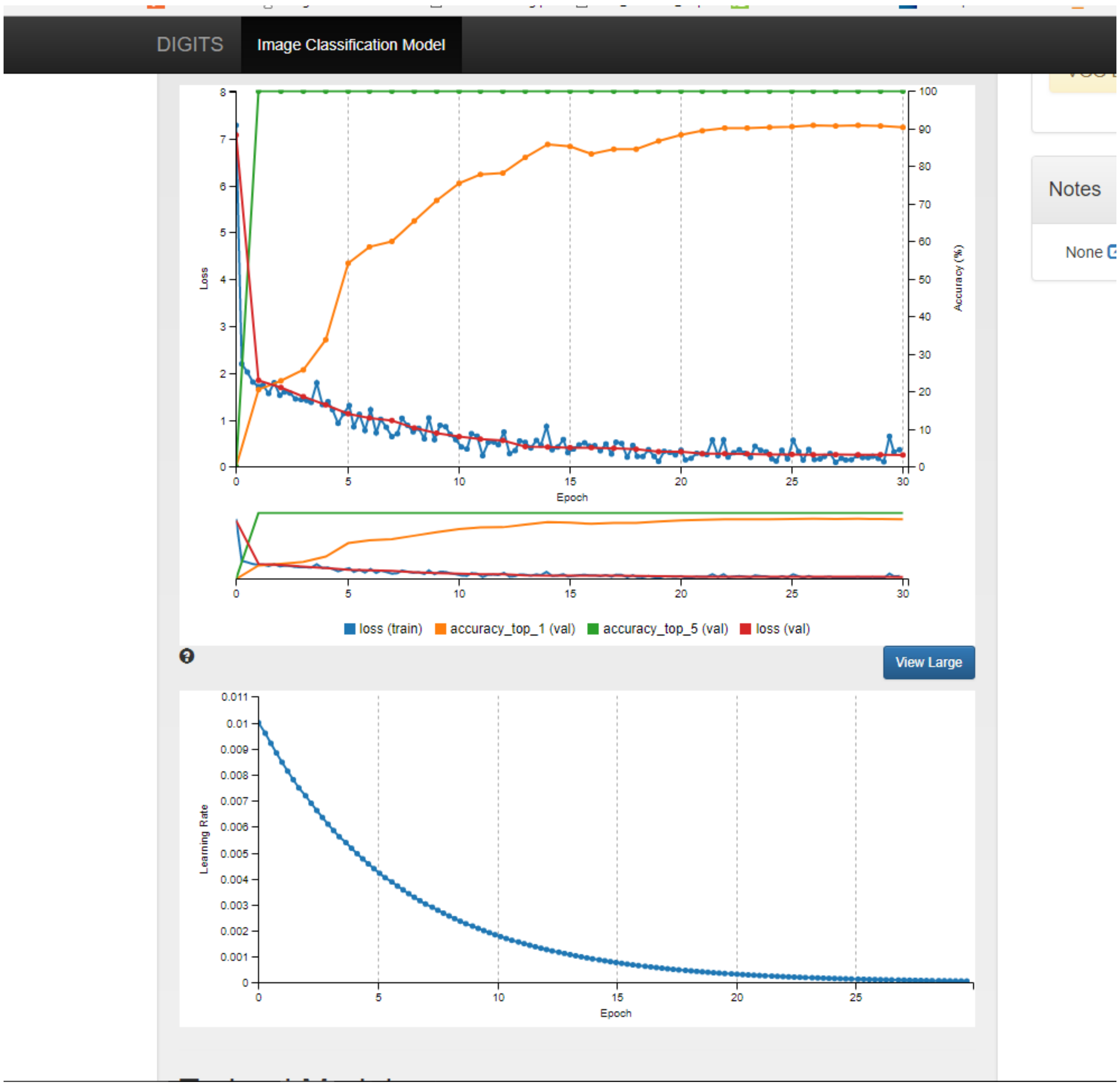
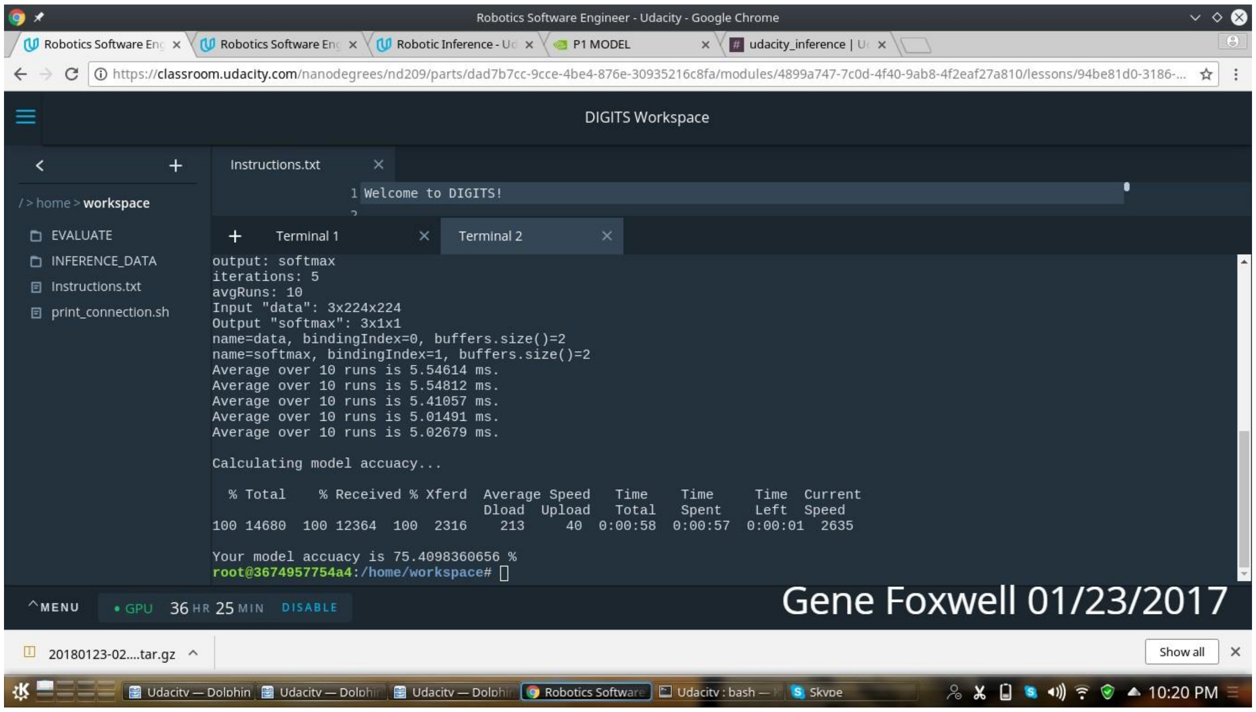


Figure 8 VGG loss curve



```
Welcome to DIGITS!
```

```
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 5.54614 ms.
Average over 10 runs is 5.54812 ms.
Average over 10 runs is 5.41057 ms.
Average over 10 runs is 5.01491 ms.
Average over 10 runs is 5.02679 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    213     40  0:00:58  0:00:57  0:00:01  2635

Your model accuacy is 75.4098360656 %
root@3674957754a4:/home/workspace# 
```

Figure 9 Udacity Dataset Performance

Despite the encouraging results shown by the confusion matrices, when run on the simulation dataset the Jetson TX2 was not quite so successful. Results from the Jetson TX2 simulation dataset are summarized in the following table.

Model	**Number of Failed Predictions**	**Final Accuracy**
GoogLeNet	126	63.2%
AlexNet	142	58.5%
VGG-16[6]	187	45.3%

Inference speed was not precisely measure during these experiments. However even without precise measurements it was easy to observe that models easily met the stated goal of one classification per second, with all models easily classifying all 342 simulation images in under 10 seconds.

DISCUSSION

While the confusion matrices for both models used in the room inference task seem promising, they were still a bit surprising. It was not expected to get near 100% (or actual 100%) per class accuracy on

[6] Note the CAFEE file for this model has not been included with this project. Its final uncompressed size exceeded 500MB, the maximum allowed size for the entire project.

the test set. It is likely that this high accuracy is caused by the limitation of the system to a single home, where there would be little deviation between instances of the living room in the training set or the test set. As there are only so many truly different perspectives that could be collected from a single room, its possible that the system was simply able to overfit to the point that nearly any applicable image from the home could be recognized for what it was.

The overfitting hypothesis is further supported by the reduced performance of the models on the simulated home walkthrough with the best model (GoogLeNet) achieving only a 63% accuracy.

There are a few possible causes for this drastic difference in performance:

1. The data for training and the data for the evaluation where collected at different times of day. It's possible that the models overfit to features that were specific to the exact time of day the data was collected. For example, the large living room windows let in vastly different levels of light during the afternoon than they do during the morning or evening hours – given the difficulty the Jetson had with confusing the Living Room with everything else its possible this was throwing off the predictions.
2. Some of the classifications where a little subjective. The nature of the home used for this experiment doesn't have 100% clearly defined borders between the Living Room, Kitchen, or Dining Room. This resulted in the experimenter using his best judgement on whether an image belonged to a room. It is possible that this was not always 100% self-consistent which would likely have contributed to the reduced Jetson TX 2 accuracy.

These issues could be mitigated by a having a much larger and more varied selection of images to train from. Collecting images from the home at many different light levels would reduce the possibility that the system is simply overfitting to the amount of reflected light it finds in the scene. This however is not practical even for one home, and may not scale to the general case.

Another approach may be to run each training image through a "pre-filter" that normalizes the lighting in each image as much as possible, reducing the amount of variance between images taken during the day and images taken during the evening.

Fixing the subjectivity of the classifications would be a little more difficult. The simplest approach here would be the crowd source the problem. Have many people look at each image and provide a single classification, then choose for the classification of the image the class that the most people voted for. This would remove the individual bias from the generated dataset (and variances caused by the same person looking at the same image at two different times and possibly coming to different conclusions) but would require far more resources than were available to the experimenter at the time of this writing.

As this projects' intended use was on generation of meta data, the actual speed of the inference engine was not a large concern. It is unlikely that a robot would find itself purposefully bouncing between rooms faster than the inference engine could provide updated meta information. Furthermore, for SLAM applications the meta data is not continuously needed – once the algorithm is reasonably sure that room R is the Kitchen, it is unlikely that this is going to morph into the bathroom.

Despite inference speed being a secondary concern to this system, it still appeared to perform reasonably well when tested on the Jetson TX2. Inference time for classifying the 342 images from the simulation set totaled to a few seconds for each model which is sufficient for most purposes. Judging by common UX practices, we would probably want the interface / robot to be able to update itself at least once a second (beyond that users start to lose patience) so the ability to classify 342 images this quickly easily exceeds that goal.

CONCLUSIONS / FUTURE WORK

An inference system was built to categorize the existing rooms of a single home. This system was tested both using a test set withheld from the original dataset, and then again using a simulated walkthrough of the home on a Jetson TX2.

On its own this project can only used in single (very specific) home with limited accuracy, however it could serve as a prototype for a much more robust system. Presuming that the challenges outlined in the discussion section are met, it is feasible that given a wider variety of well classified room data that this approach could be extended to reasonable well in most existing homes. (It may turn out to be the case that localizations of the model are required to account for high variance in room structure and furniture between different cultures). Future would then start by enhancing the data collection and labeling techniques used and training a more robust model.

Once the inference engine has been extended there are many possible commercial applications. These include (but may not be limited to):

- Augmented Reality – this system could adjust the data / objects being displayed based on the room the user is in.
- Home Robotics – A home robot that can create and understand plans like “go to the living room and get me the remote control” would benefit greatly from being able to automatically confirm it is in fact in the living room.
- SLAM – This system could be used to automatically add meta data to maps being created from laser scanners or other such devices. Such a system might be used by a real-estate agent for example attempting to provide a way for remote buyers to view or inspect a potential new property. Additional, being able to identify what part of an environment the robot is currently in could also be used to help with exploration. If the robot knows its looking for the remote control, and knows the remote is in the living room – then it also knows it needs to find a living room first before looking for the remote.