Robotic Inference

Christopher Ohara

Abstract—Object detection and classification are topics becoming more pertinent for roboticists, as machines begin taking on tasks that have traditionally been considered manual labor. Until recently, object classification via sensors and active robotic inference has been a difficult task, especially in stochastic and non-predictable environments. Advances in convolutional neural networks have lead to more pragmatic and realistic applications for companies like Siemens, for developing smart factories. In this project, two different neural networks architectures utilizing a variety of optimizers are explored for the classification of two datasets; a standard factory conveyor belt utilizing a Jetson TX2 and a dataset consisting of drones.

Index Terms—Robotics, Inference, Udacity, Object Detection, DNN, Jetson TX2, Deep-learning, Drone, UAV, Classification.

1 Introduction

Since 2010, the ImageNet Challenge has encouraged researches to join competitions that are intended to advance the fields of computer vision and machine learning through neural networks and novel architectures. In 2012, a breakthrough was achieved in when AlexNet was created and utilized to revolutionize deep learning by achieving unprecedented results. AlexNet, becoming nearly a de facto standard, was programmed with CUDA to run quickly on Nvidia GPUs. [1]

Robotic inference has been gaining more traction since this time, as real-world implementations are becoming more realizable. Smart factories are now able to categorize items based on their labeled data from segmentation and classification models. Combined with real-time devices designed for machine learning and artificial intelligence, like the Jetson TX2, objects can be detected and classified very quickly (in the milliseconds) and with a high accuracy given the proper network architecture and hyperparameters. [2]

Not only AlexNet is explored, as the newer GoogLeNet architecture is also implemented. Three different optimizers are also utilized, including Adam, Stochastic Gradient Descent (SGD), and AdaGrad (an extended version of SGD).

2 Data Acquisition

The project has two distinct goals. The first goal is to analyze and benchmark a dataset provided by Udacity using Nvidia DIGITS. The requirements are to define a model that is able to return an inference speed of less than 10ms and have an accuracy of greater than 75%. The second portion of the project is to create a person dataset and model, evaluating design choices and comparative results.

For the first phase of the project, a dataset was provided through the DIGITS environment. The dataset included candy boxes, bottles, and images of ”nothing”, that were run through a conveyor belt to check for the accuracy of the object detection and recognition sensors. These images were taken with a Jetson TX2 development kit. The dataset was split into two sets of images when developing the model; a training set and a validation set. The training set consisted of 7570 images, whereas the validation set contained 2523 images (25%).

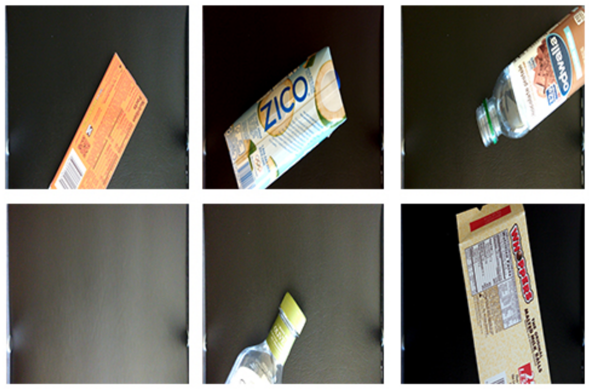


Fig. 1: Example of Supplied Dataset Images.

The second dataset was personally created with 646 images were taken of four distinct UAV-related objects. Approximately 120 images of each item were taken for the testing set, and 40 images of each were taken for the validation set. These objects are of Parrot Drone products, including the Bebop 2, SkyController 1, and AR.Drone 2.0 (bare version and winter camouflage version). The intention behind the object selection was taken with the intent to classify autonomous vehicles, as they are becoming more prominent each day, even though there is a lack of available datasets for these flying embedded devices.

The images have the dimensions of 256x256 pixels, which is required for both the AlexNet and GoogLeNet CNN architectures as inputs. Images for the personal dataset were taken with a 1:1 ratio, with 4.3 mega-pixels using a Samsung S9+ smart phone. This allowed for rapidly acquiring high definition images at a 1:1 ratio that prevents the squishing of data during transformation. The images were taken at 2160x2160 pixels, and then a Python script using OpenCV was used to resize the images.

The DIGITS hardware used is a Tesla K80. The K80 is a GPUGP card Accelerator containing 4992 CUDA cores delivering up to 2.91 teraflops double-precision performance with NVIDIA GPU Boost. As a streaming processor, it is able to quickly compile data through parallelization. [3]

3 Network Choices

The first model designed was on the GoogLeNet architecture utilizing the Adam optimizer. The table below characterizes the criteria selection. Adam was chosen as it has low memory requirements, Adam computes adaptive learning rates for parameters. Adam stores an exponentially decaying average of past squared gradient as well as an exponentially decaying average of past gradients. [4]

TABLE 1: DIGITS Dataset: GoogLeNet w/ Adam - 45.17% Accuracy, 10 Epochs.

Solver Option	Value
Architecture	GoogLeNet
Epochs	10
Snapshop Interval	1.0
Validation Interval	1.0
Batch Size	128
Solver Type	Adam
Learning Rate	0.01
Policy	Step Down
Step Size	33.0
Gamma	0.1
Accuracy	45.17%

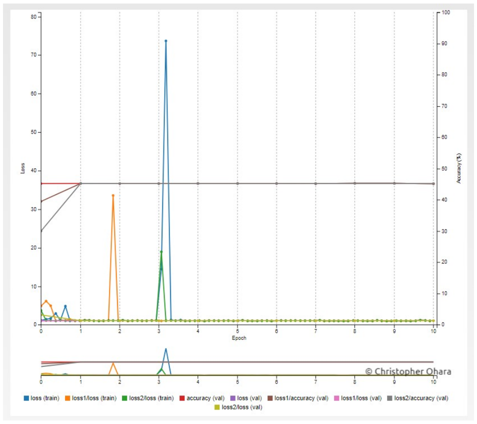


Fig. 2: DIGITS Dataset: GoogLeNet w/ Adam - 45.17% Accuracy, 10 Epochs.

However, as can be seen from above, Adam had some difficultly in returning high accuracy for the dataset. As the dataset is rather large, and the default learning policy is to use a step size, Adam would not have improved by increasing the number of epochs. The learning rate and batch size could have been tuned to improve results, but it was decided to reattempt with another optimizer.

One of the most well-renown optimized is SGD. SGD can very quickly arrive at an optimal minimum, which allows the accuracy to be computed very quickly as seen below. In only two epochs, the target accuracy has already been reached and is steady-state. The only specifications that were altered was changing the optimizer to SGD.

For the personal dataset, AlexNet with AdaGrad was selected. AdaGrad combined SGD with smaller learning rates

TABLE 2: DIGITS Dataset: GoogLeNet w/ SGD - 99.88% Accuracy, 10 Epochs.

Solver Option	Value
Architecture	GoogLeNet
Epochs	10
Snapshop Interval	1.0
Validation Interval	1.0
Batch Size	128
Solver Type	SGD
Learning Rate	0.01
Policy	Step Down
Step Size	33.0
Gamma	0.1
Accuracy	99.88%

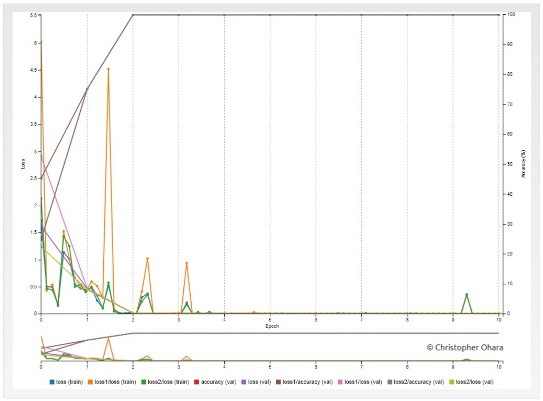


Fig. 3: DIGITS Dataset: GoogLeNet w/ SGD - 99.88% Accuracy, 10 Epochs.

and is ideal for sparse data. Since the images contain a lot of empty space (white and black backgrounds), this optimizer would be ideal for training the network. However, since this is a relatively small dataset, running only 30 epochs did not yield the desired results. Once the second attempt, the epoch number was doubled, allowing the images to run through the network twice as many times. This yielded much more favorable results, as will be described below.

TABLE 3: Personal Dataset: AlexNet w/ AdaGrad - 79.55% Accuracy, 30 Epochs.

Solver Option	Value
Architecture	AlexNet
Epochs	10
Snapshop Interval	1.0
Validation Interval	1.0
Batch Size	128
Solver Type	AdaGrad
Learning Rate	0.01
Policy	Exp. Decay
Gamma	0.95
Accuracy	79.55%

As can be seen in the personal dataset table and image below, only increasing the number of epochs drastically changed the results. A habit of introductory machine learners and roboticists is to arbitrarily change multiple values and hyperparameters simultaneously, treating the system as a black box. However, neural network behavior is becoming

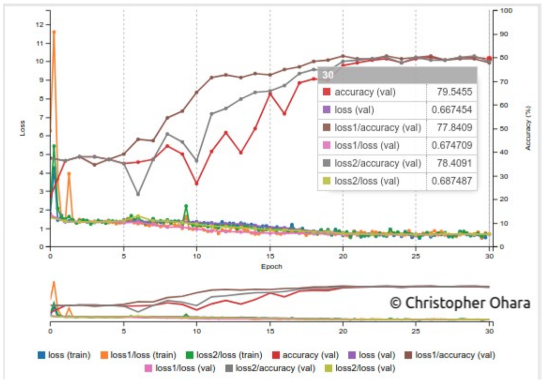


Fig. 4: Personal Dataset: AlexNet w/ AdaGrad - 79.55% Accuracy, 30 Epochs.

more predictable and composable, so design choices should be justified early (usually depending on available hardware and computing services), and only updated in a logical manner.

TABLE 4: Personal Dataset: AlexNet w/ AdaGrad - 90.35% Accuracy, 60 Epochs.

Solver Option	Value
Architecture	AlexNet
Epochs	10
Snapshop Interval	1.0
Validation Interval	1.0
Batch Size	128
Solver Type	AdaGrad
Learning Rate	0.01
Policy	Exp. Decay
Gamma	0.95
Accuracy	90.35%

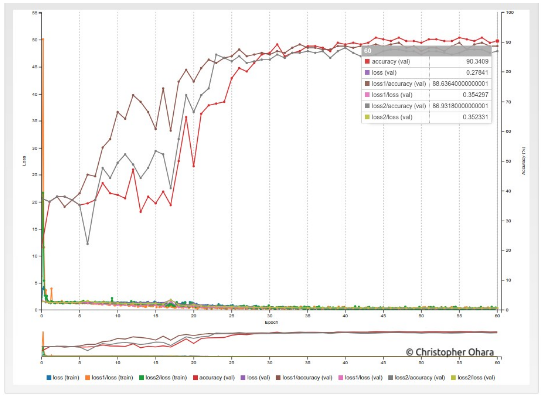


Fig. 5: Personal Dataset: AlexNet w/ AdaGrad - 90.35% Accuracy, 60 Epochs.

However, since all of these images have already been taken and transformed, it would be logically to see what the SGD optimizer can do achieve.

4 Results

Once the accuracy of the model reached nearly 100%, a few random images were selected to validate the model.

TABLE 5: Personal Dataset: AlexNet w/ SGD - 98.15% Accuracy, 60 Epochs.

Solver Option	Value
Architecture	AlexNet
Epochs	10
Snapshop Interval	1.0
Validation Interval	1.0
Batch Size	128
Solver Type	SGD
Learning Rate	0.01
Policy	Exp. Decay
Gamma	0.95
Accuracy	98.15%%

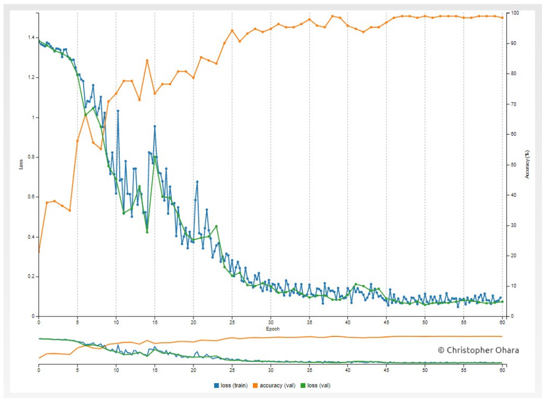
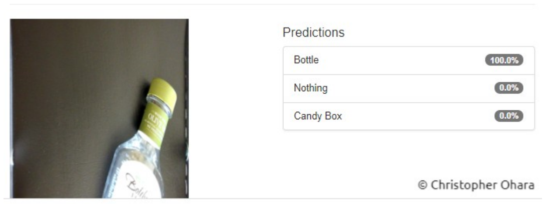


Fig. 6: Personal Dataset: AlexNet w/ SGD - 98.15% Accuracy, 60 Epochs.

As expected, the classifier was able to correctly distinguish between bottles, candy boxes, and ”nothingness.”

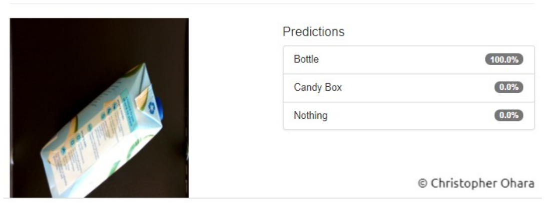


Predictions

Bottle	100.0%
Nothing	0.0%
Candy Box	0.0%

© Christopher Ohara

Fig. 7: Bottle - 100.00% Accuracy.



Predictions

Bottle	100.0%
Candy Box	0.0%
Nothing	0.0%

© Christopher Ohara

Fig. 8: Bottle - 100.00% Accuracy.

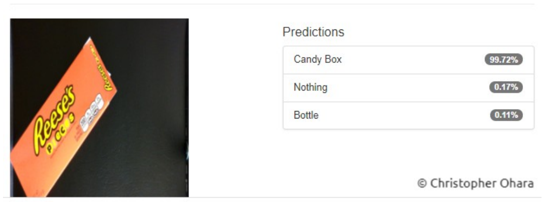

Predictions
Candy Box 99.72%
Nothing 0.17%
Bottle 0.11%
© Christopher Ohara

Fig. 9: Candy Box - 99.72% Accuracy.

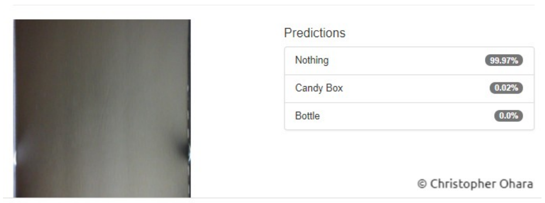
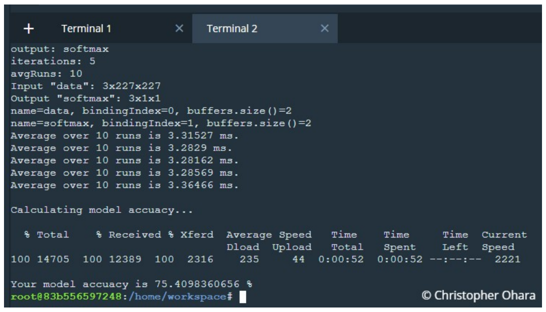

Predictions
Nothing 99.97%
Candy Box 0.02%
Bottle 0.0%
© Christopher Ohara

Fig. 10: Nothing - 99.97% Accuracy.

After the model has been trained and analyzed, it is sent back to the DIGITS workspace. This allows for testing the robotic inference rate. Specifically, the inference device utilizes Tensor RT 3.0 to quickly average the speed for ten attempts over five runs. As a result, the inference speed was around 3.31ms with an accuracy of 75%.


```
Terminal 1    Terminal 2
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 3.31527 ms.
Average over 10 runs is 3.2829 ms.
Average over 10 runs is 3.28162 ms.
Average over 10 runs is 3.28569 ms.
Average over 10 runs is 3.36466 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14705  100 12389  100  2316    235     44  0:00:52  0:00:52 --:--:--  2221

Your model accuacy is 75.4098360656 %
root@83b556597248:/home/workspace#
```
© Christopher Ohara

Fig. 11: Jetson Inference: Speed_{ave} = 3.31ms, & 75.41% Accuracy.

Sending the dataset through 30 epochs was very quick, taking less then three minutes. However, the results were not likely to be satisfactory. To test the hypothesis, several images were randomly selected from the validation set. As can be seen, the images lack a high confidence value. The images shown below represent the bare AR.Drone 2.0 model, as it is the most challenging for humans to differentiate as well.

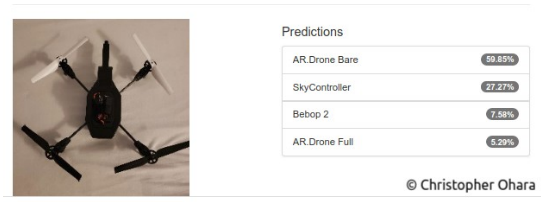

Predictions
AR.Drone Bare 59.85%
SkyController 27.27%
Bebop 2 7.58%
AR.Drone Full 5.29%
© Christopher Ohara

Fig. 12: AR.Drone Bare - 59.85% Accuracy.

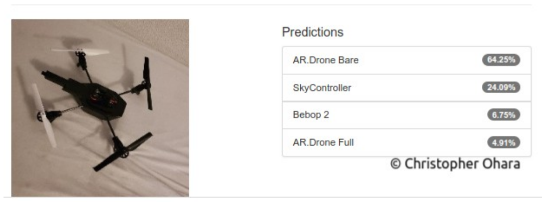

Predictions
AR.Drone Bare 64.25%
SkyController 24.09%
Bebop 2 6.75%
AR.Drone Full 4.91%
© Christopher Ohara

Fig. 13: AR.Drone Bare - 64.25% Accuracy.

Without adding more images to the dataset or changing parameters other than the epoch number, the accuracy drastically improved. The SkyController and bare AR.Drone were the most difficult items to distinguish, especially on the dark background. However, the Bebop 2 and AR.Drone with the snow camouflage cover exceeded 99% accuracy with a white background, thanks to AdaGrad and GoogLeNet. This was with only two hundred images of each device as well.

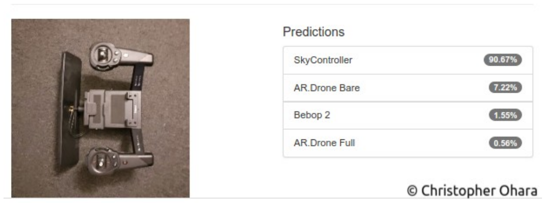

Predictions
SkyController 90.67%
AR.Drone Bare 7.22%
Bebop 2 1.55%
AR.Drone Full 0.56%
© Christopher Ohara

Fig. 14: SkyController - 90.67% Accuracy.

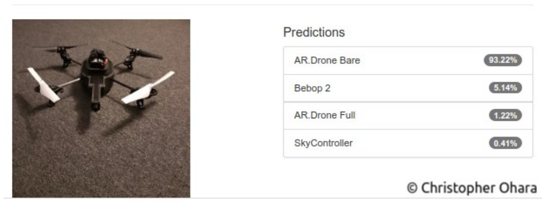

Predictions
AR.Drone Bare 93.22%
Bebop 2 5.14%
AR.Drone Full 1.22%
SkyController 0.41%
© Christopher Ohara

Fig. 15: AR.Drone Bare - 93.22% Accuracy.

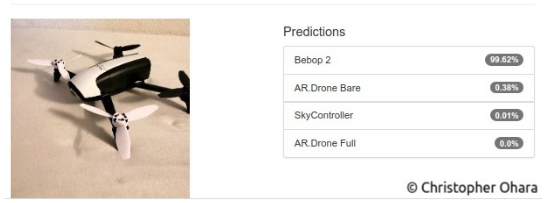

Predictions
Bebop 2 99.62%
AR.Drone Bare 0.38%
SkyController 0.01%
AR.Drone Full 0.0%
© Christopher Ohara

Fig. 16: Bebop 2 - 99.62% Accuracy.

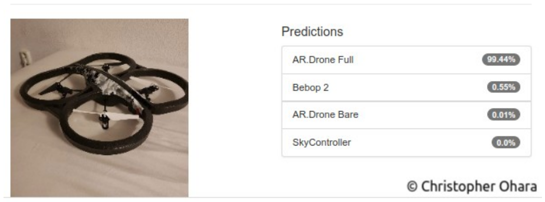
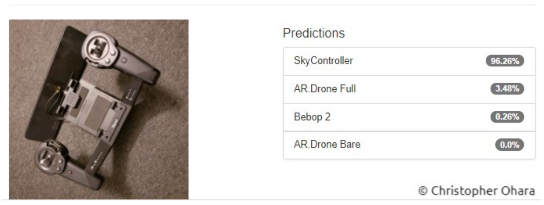

Predictions
AR.Drone Full 99.44%
Bebop 2 0.55%
AR.Drone Bare 0.01%
SkyController 0.0%
© Christopher Ohara

Fig. 17: AR.Drone Full - 99.44% Accuracy.

As can be seen below, using SGD has still outperformed the other more ”sophisticated” optimizers. Ockham’s Razor states that ”between two (or more) equally likely options, the simplest one should be chosen.” This is an appropriate demonstration of justifying this approach, as SGD is usually set as the default optimizer, works quickly, and provides wonderful results.



Predictions

SkyController	96.26%
AR.Drone Full	3.48%
Bebop 2	0.26%
AR.Drone Bare	0.0%

Fig. 18: SkyController - 96.26% Accuracy.

5 Discussion

Overall, the results of both phases of the project yielded excellent results. Having access to a large dataset will assist in ensuring accurate classification and segmentation, at a cost of requiring large amount of memory (data), increased computational requirements, and increased energy consumption. On an embedded platform like the Jetson TX2, it is not ideal to spend too much effort energy for object recognition and detection. However, if time is not constrained (soft real-time or non-critical) or energy/memory is not an issue (hardwired, host system), then it is ideal to maximize the accuracy. The second phase of the project has show that neural networks can be trained quickly and effectively, even with a dataset that was only approximately 6% of the size. A smaller dataset also allows for more flexibility in experimentation/application, as the results can be analyzed quickly.

6 Conclusion / Future work

Due to the increasing capabilities of deep-learning and machine learning, real-world applications of inference will continue to expand until they are seamlessly integrated into our daily lives. There is a vast amount of flexibility in design criteria, that can be altered to meet functional and quality requirements. For future work, a real-time image acquisition and deep-learning classification network will be designed utilizing the Jetson TX2 and the Parrot Bebop 2. The goal will be to find the minimal dataset size to yield ”just barely good enough” (JBGE) results when classifying targets and objects of interest for UAVs.

References

[1] http://fortune.com/ai-artificial-intelligence-deep-machine-learning/
[2] https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf
[3] https://www.nvidia.com/en-us/data-center/tesla-k80/
[4] http://ruder.io/optimizing-gradient-descent/index.html