Julian Singkham
CS 2300/11
Dr. Berisha

# Lab 8: Self-Driving Car DNN

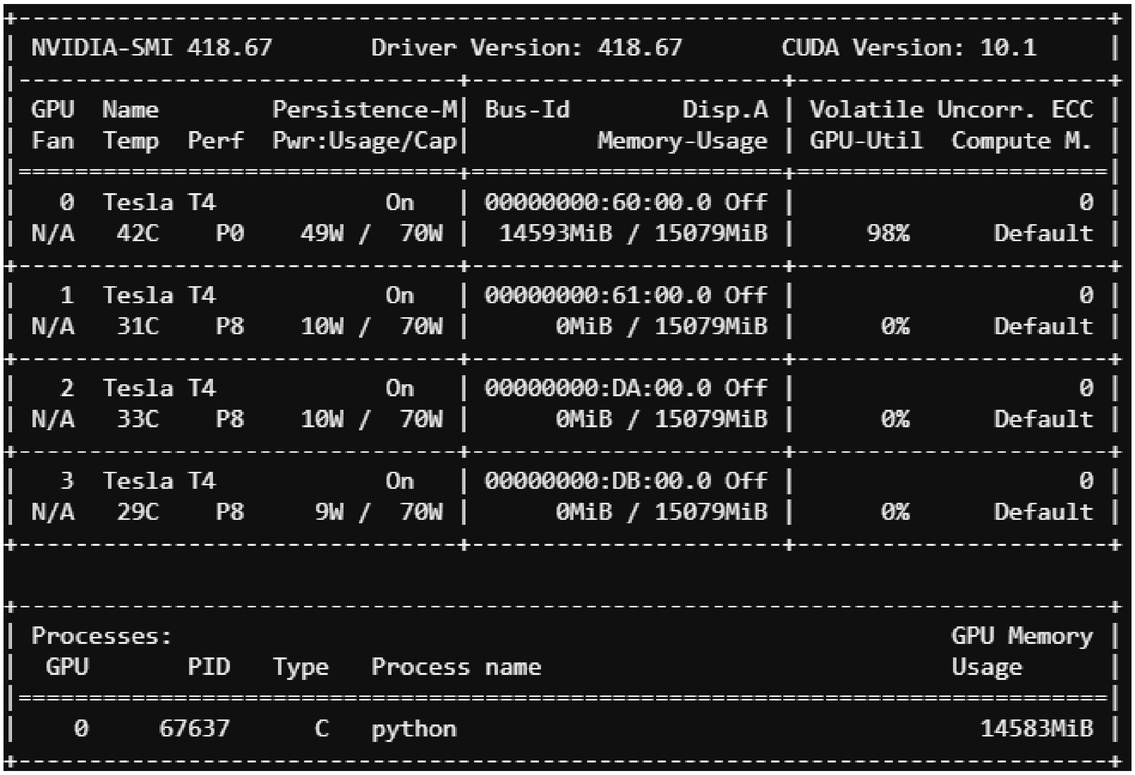
## Abstract

The purpose of this lab is to familiarize ourselves with running batch jobs on Rosie and finding the optimal parameters to get a minimum of 98% accuracy on fresh/rotten fruit detection.

The dataset used for this lab is ‘Fruits Fresh and Rotten for Classification” by Sriram Reddy Kalluri. The dataset contains 13.6k (1.82 GB) .png images of apples, oranges, bananas, rotten apples, rotten oranges, rotten bananas.

The provided Lap9.py has the following paramters: data, batch_size, epochs, main_dir, augment_data, fine_tune

## Part 1: Training interactively via the command line in a container



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.67       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:60:00.0 Off |                    0 |
| N/A   42C    P0    49W /  70W |  14593MiB / 15079MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            On   | 00000000:61:00.0 Off |                    0 |
| N/A   31C    P8    10W /  70W |      0MiB / 15079MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla T4            On   | 00000000:DA:00.0 Off |                    0 |
| N/A   33C    P8    10W /  70W |      0MiB / 15079MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla T4            On   | 00000000:DB:00.0 Off |                    0 |
| N/A   29C    P8     9W /  70W |      0MiB / 15079MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     67637      C   python                                     14583MiB |
+-----------------------------------------------------------------------------+
```

*Figure 1: GPU Usage Data*

To determine which GPU is mine, I first have to find the PID of my process. Since I am the only one on this node, there is only one process running which is on GPU 0. If there were more, I would have to run **ps -o user= -p PIDHERE** on each PID to see who the owner is.

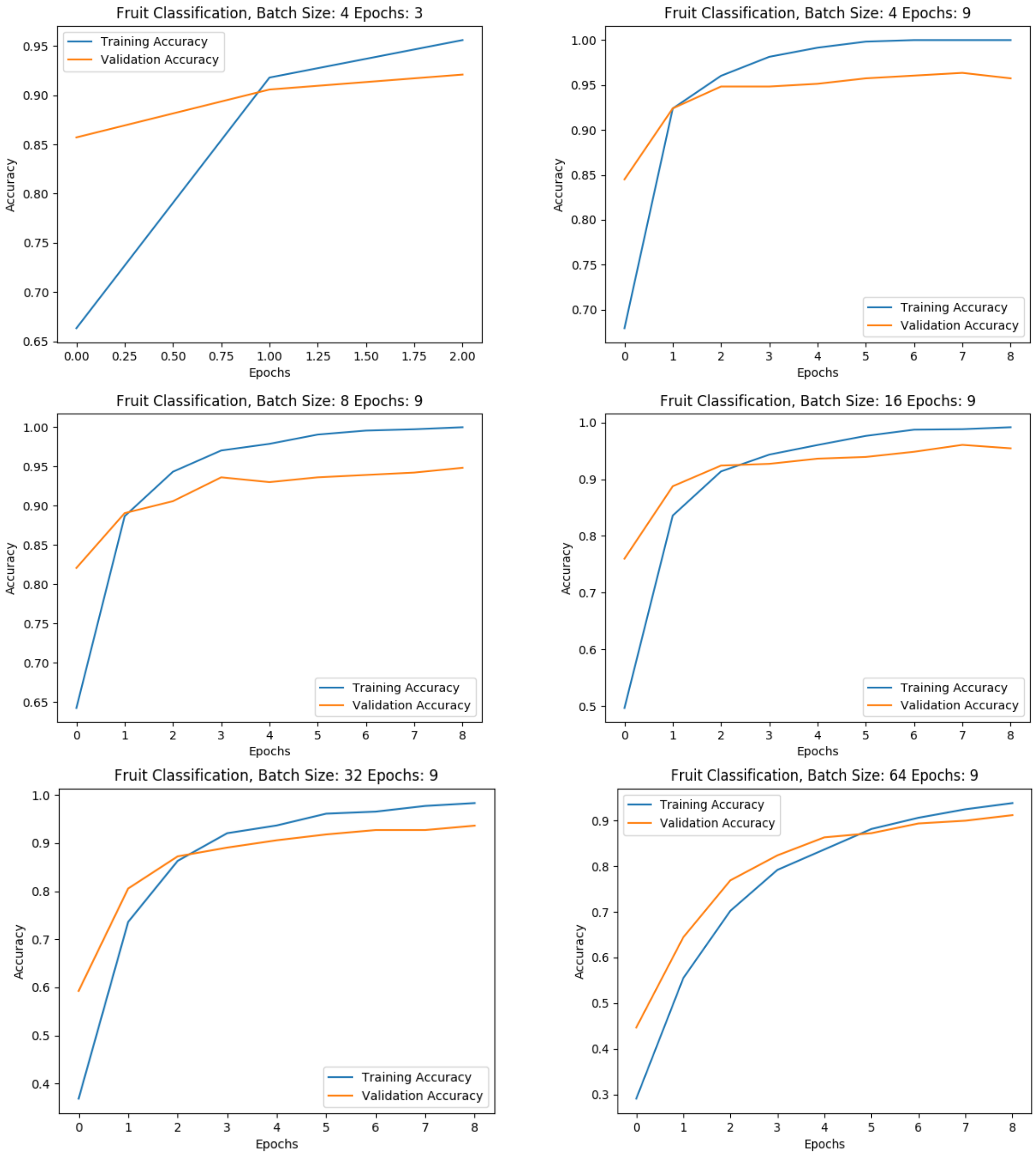
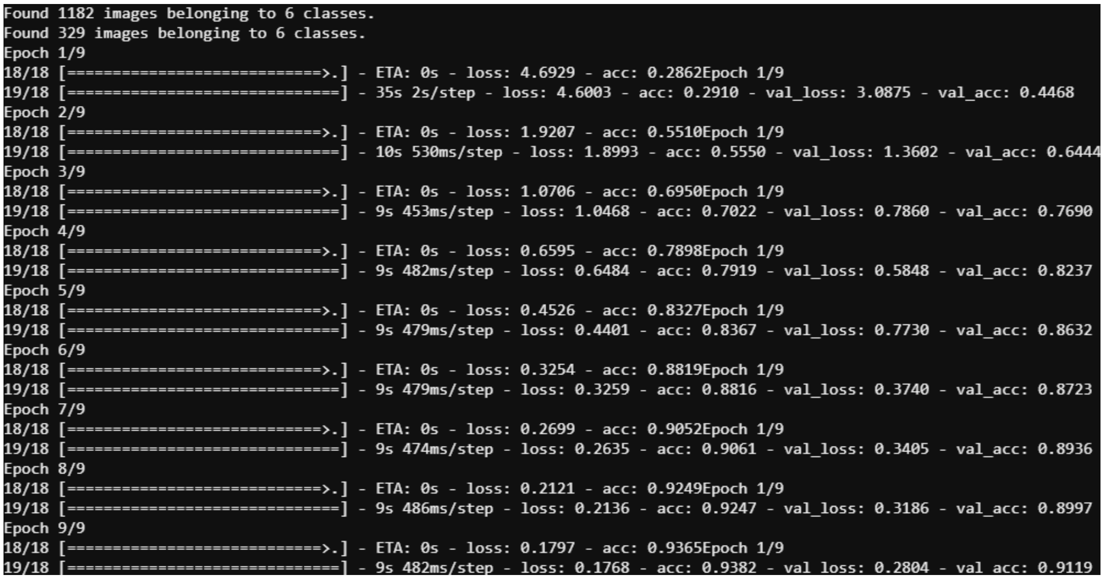


*Figure 2: Fruit Classification with Different Batch Sizes and Epochs*

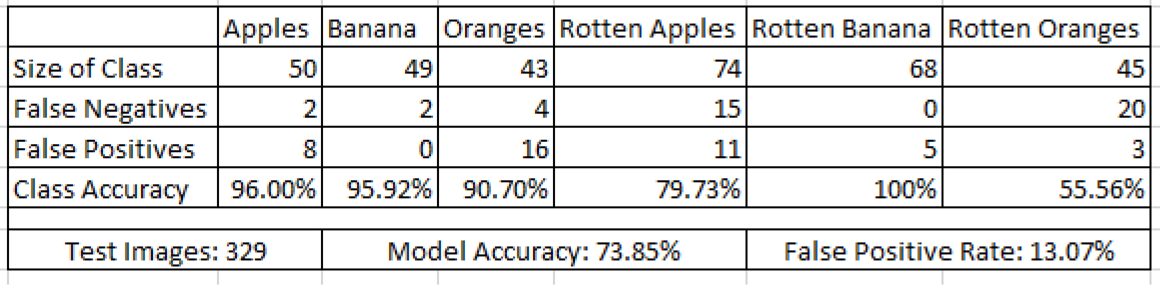
Increasing the Epoch had minimal improvements in accuracy as overfitting occurred at epoch 3. The highest recorded accuracy occurred with a batch size of 16 and epoch 2 (~92%). Increasing the batch size past 16 lead to an overall decrease in accuracy; ~2% at batch size 32 and ~5% at batch size 64.



```
Found 1182 images belonging to 6 classes.
Found 329 images belonging to 6 classes.
Epoch 1/9
18/18 [============================>.] - ETA: 0s - loss: 4.6929 - acc: 0.2862Epoch 1/9
19/18 [==============================] - 35s 2s/step - loss: 4.6003 - acc: 0.2910 - val_loss: 3.0875 - val_acc: 0.4468
Epoch 2/9
18/18 [============================>.] - ETA: 0s - loss: 1.9207 - acc: 0.5510Epoch 1/9
19/18 [==============================] - 10s 530ms/step - loss: 1.8993 - acc: 0.5550 - val_loss: 1.3602 - val_acc: 0.6444
Epoch 3/9
18/18 [============================>.] - ETA: 0s - loss: 1.0706 - acc: 0.6950Epoch 1/9
19/18 [==============================] - 9s 453ms/step - loss: 1.0468 - acc: 0.7022 - val_loss: 0.7860 - val_acc: 0.7690
Epoch 4/9
18/18 [============================>.] - ETA: 0s - loss: 0.6595 - acc: 0.7898Epoch 1/9
19/18 [==============================] - 9s 482ms/step - loss: 0.6484 - acc: 0.7919 - val_loss: 0.5848 - val_acc: 0.8237
Epoch 5/9
18/18 [============================>.] - ETA: 0s - loss: 0.4526 - acc: 0.8327Epoch 1/9
19/18 [==============================] - 9s 479ms/step - loss: 0.4401 - acc: 0.8367 - val_loss: 0.7730 - val_acc: 0.8632
Epoch 6/9
18/18 [============================>.] - ETA: 0s - loss: 0.3254 - acc: 0.8819Epoch 1/9
19/18 [==============================] - 9s 479ms/step - loss: 0.3259 - acc: 0.8816 - val_loss: 0.3740 - val_acc: 0.8723
Epoch 7/9
18/18 [============================>.] - ETA: 0s - loss: 0.2699 - acc: 0.9052Epoch 1/9
19/18 [==============================] - 9s 474ms/step - loss: 0.2635 - acc: 0.9061 - val_loss: 0.3405 - val_acc: 0.8936
Epoch 8/9
18/18 [============================>.] - ETA: 0s - loss: 0.2121 - acc: 0.9249Epoch 1/9
19/18 [==============================] - 9s 486ms/step - loss: 0.2136 - acc: 0.9247 - val_loss: 0.3186 - val_acc: 0.8997
Epoch 9/9
18/18 [============================>.] - ETA: 0s - loss: 0.1797 - acc: 0.9365Epoch 1/9
19/18 [==============================] - 9s 482ms/step - loss: 0.1768 - acc: 0.9382 - val_loss: 0.2804 - val_acc: 0.9119
```

*Figure 3: Fruit Classification with batch size of 64*

**Part 2: Testing model on validation images.**



| | Apples | Banana | Oranges | Rotten Apples | Rotten Banana | Rotten Oranges |
|---|---|---|---|---|---|---|
| Size of Class | 50 | 49 | 43 | 74 | 68 | 45 |
| False Negatives | 2 | 2 | 4 | 15 | 0 | 20 |
| False Positives | 8 | 0 | 16 | 11 | 5 | 3 |
| Class Accuracy | 96.00% | 95.92% | 90.70% | 79.73% | 100% | 55.56% |
| | | | | | | |
| Test Images: 329 | | Model Accuracy: 73.85% | | | False Positive Rate: 13.07% | |

*Table 1: Fruit Classification Validation Accuracy*

The two types of misclassifications are:

- False Negative: Predicted as false but answer is true.
- False Positive: Predicted as positive but answer is false.

Rotten oranges by far had the worst accuracy with 55.56% and it deviated the most from the other classes. Overall the model has poor accuracy at 73.85% and a false positive rate of 13.07%. False positives are the most dangerous as they allow false data to pass as true. For instance, if this model was used to filter produce for a super market, having rotten fruit pass as true could endanger the consumers.

## Part 3: Batch mode

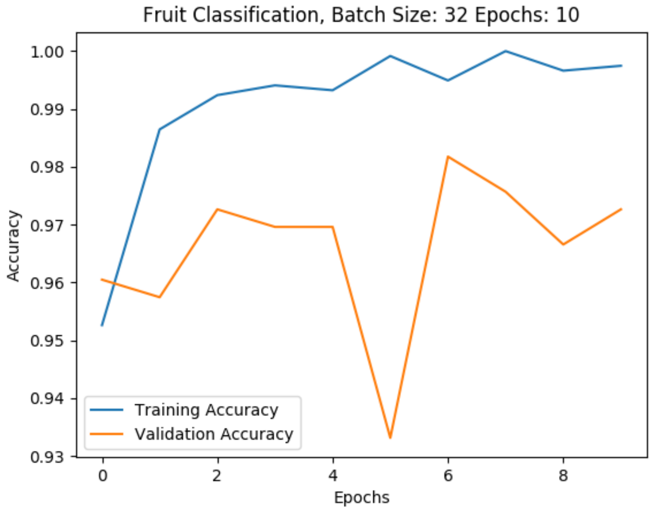


*Figure 4: Fruit Classification Batch method*

The batch job was frustrating for me as I didn't understand the format of the shell file beforehand. One of the biggest problems is that I removed the #'d commands as I assumed, they were left as comments, so I knew to edit them. As it turns out #'s are needed for the shell file to work. Additionally, I had my lab under "Lab 9", which caused problems as the shell file couldn't understand why I had a space in it. The shell file had a very long waiting time, so I ended up spending an hour to debug it. Beyond that the shell approach is efficient as you can run 1 very short command instead of running the very long command used in Part 1. Furthermore, you can edit the shell commands in an editor while just using bash has issues (copy pasting and having to use arrow keys to navigate). Another advantage is that the asynchronous nature of batch mode is you can schedule multiple jobs at once which allows for one to queue a bunch of jobs and then let it run overnight. The only disadvantage I could think of is that the queueing time is at the whim of the Rosie scheduler whereas the alternative has you request a block of time on Rosie, and once you have a block of time you are guaranteed that time. This means there could be a large wait time between batch jobs.

CPU usage can be seen by SSHing into the node the batch file is running in and calling the **top** command. Increasing the CPU count from 2 to 18 has no affect on performance, as the cpu usage hovers at 100% in either cases. The max seen usage is at 191%.

## Part 4: Hyperparameters

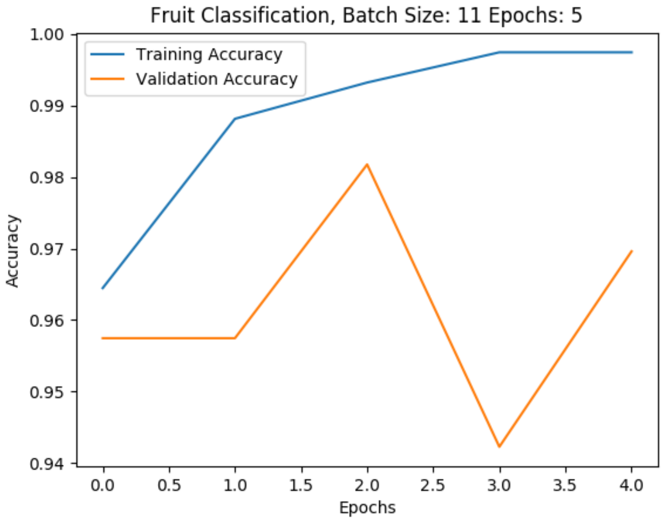


*Figure 5: Fruit Classification Optimal Parameters*

To achieve an accuracy of 98% I halved the learning rate, 0.00001 to 0.000005, epoch of 5, and ran batch sizes of 10-20 as batch 16 in Figure 2 had the highest level of accuracy. I changed the learning rate as learning rate plays a large part in the accuracy level and I believed that would have the highest effect on increasing the accuracy to 98%.