WGU C964

Warren Harper

Date: 07/30/2021

NOTE: Before reviewing this submission I ask that you watch the video I made to showcase this project. I made this video due to the physical nature of this project and not being able to send you, the evaluator, the physical car to play with.

https://www.youtube.com/watch?v=Y7Xbld2h_E8

If you were to have the physical car, you would run **arnie.py** from the raspberry pi command line. **Arnie.py** is the main program to review as this brings all the elements together including the 2 machine learning models that I trained in order for the car to function.

All the source files can be found on my github or google drive:

https://github.com/wnharper/arnie

https://drive.google.com/drive/folders/1EHQbNcWBfqyHrlfVpORqQSo3kfvstsv_?usp=sharing

There are also two Jupyter notebook files. I recommend viewing these through github as you will be able to view my last execution of each cell. These are not designed to be run by an evaluator as they require a sync to my google drive, and multiple runtime restarts. I used Google's Collab to create them and train the models (which took over 17 hours). On **page 27** I also provide an explanation for each program/file found in github/google drive.

I really wanted to push myself and do something different. I hope you enjoy evaluating this as much as I did creating it.

WGU C964

Task 2 - Section A

LETTER OF TRANSMITTAL

Warren Harper

The Zee Car Company
410 Auto way
CA, 95114

X-Soft
1 Tensor way
CA, 92672

To whom it may concern:

Zee Car Company (ZCC) is looking to automate their parts delivery system within their factories. We at X-Soft recommend a self driving delivery vehicle that can navigate the factory floor autonomously and deliver parts as needed.

ZCC will save money in staffing costs as the vehicle will be able to operate 24 hours a day without explicit supervision. The vehicle will also be able to make deliveries faster and more efficiently than the current method which is by hand. ZCC will not need to install any major infrastructure to accommodate the self-driving vehicle other than some painted lines on the factory floor.

The primary objective of this project is to create a program that will enable the delivery vehicle to drive autonomously. The program should be easy to use, maintain, and able to adapt to the various ZCC factory layouts and locations.

The total cost for this project will be approximately $58,000 for software development and $92,000 for the physical car. This does not include any ongoing maintenance.

X-Soft is the industry leader in golf cart automation and we believe we have the expertise to complete this project. X-Soft would be thrilled to partner with ZCC on this ground breaking project.

Kind Regards

Warren Harper

CEO

Task 2 - Section A

PROJECT RECOMMENDATION

Problem Summary

Zee Car Company (ZCC) is looking for a way to deliver parts autonomously from the onsite warehouse to various locations on the factory floor and campus. The factory runs 24 hours a day and so ZCC needs a solution that does not require explicit supervision by a human (employee).

ZCC would like a solution that requires minimal infrastructure change and a solution that can either adapt or easily change to suit multiple factory layouts.

The proposed solution is a self-driving delivery vehicle (electric) that can navigate a set of painted lanes. The solution will be able to adapt to different routes set out using the painted lane method.

The scope of this solution includes all software development, the training of the machine learning models in order for the vehicle to navigate autonomously, and one physical delivery vehicle with software installed. It does include multiple physical vehicles, software licenses for more than one vehicle, and ongoing maintenance to the software or vehicle.

Application benefits

ZCC currently relies on delivering parts by hand from it's onsite warehouse to various locations on the factory floor. This process requires staffing 24 hours a day. It is also slow and inefficient. ZCC does not want to invest in permanent infrastructure, such as tracks, in order to deliver the parts as the factory layout is prone to changing and they have multiple factories with different layouts.

The proposed solution will be completely automated and will be able to operate unsupervised. It also does not require any significant infrastructure. The only requirement is a painted lane on the factory floor. The routes and the factory floor layout can be

changed and the self driving vehicle will be able to adapt. Once the pilot program has been implemented, it can be rolled out to ZCC's other factories with relative ease.

Application description

The proposed solution will be a fully electric delivery vehicle which will use a battery as its main source of power. This is both quiet and environmentally friendly especially in a factory environment.

It will carry an onboard computer capable of processing images through two machine learning models in real-time. The first machine learning model will enable the vehicle to navigate a painted lane on a factory floor. The vehicle will be able to navigate unseen routes. The second machine learning model will allow the vehicle to detect objects in close proximity and make decisions based on those detected objects. These objects include specially designed signs, for example a speed limit sign, and other potential hazards or persons.

Data description

The data that will be used in order to train the models will be collected and processed by X-Soft. X-Soft will construct training routes (painted lanes) inside environments similar to that of a ZCC factory and use a vehicle similar to that of the final solution to gather information as it is driven by hand through various training routes and environments.

X-Soft will also collect, process and produce the images needed in order to train the object detection model.

Objective and Hypotheses

The objective of this project is to produce a fully autonomous vehicle that can deliver parts to various locations within a ZCC factory.

If the proposed solution is able to navigate an unseen path without deviating from the path, and the proposed solution is able to detect predetermined objects in close proximity and make decisions based on those detected objects, then the proposed solution will be able to operate autonomously, therefore fulfilling the primary objective.

Methodology

Development will follow the SEMMA methodology. Sample, Explore, Modify, Model, Asses.

This methodology is particularly effective for data modelling which will make up the majority of the proposed solution. We believe that a self-driving vehicle needs to have a 100% success rate and therefore the modelling process has to be extremely rigorous. We can not deploy the vehicle until it is able to navigate the designated lanes and detect particular objects without failure.

Funding requirements

The total cost for this project will be approximately $58,000 for software development and $92,000 for one physical vehicle. This does not include any ongoing maintenance of the software or the physical vehicle. The cost does not include extra software licenses for additional vehicles.

Stakeholders impact

The proposed solution will have a direct impact on the parts fulfillment office. The method by which they will deliver parts will be completely changed.

Any parties on the factory floor will be directly or indirectly affected by the vehicle. The manner in which various departments receive parts will be affected.

Human resources will be affected as certain functions within the organization will be rendered obsolete.

Data precautions

There will be no collection or storage of sensitive data. When the vehicle is in the final stages of training, there will be images collected of the factory floor (with painted lines). Although the camera is orientated in such a way that it could potentially take photographs of sensitive material (within the factory), once the images are processed for machine learning purposes, there will be no sensitive material visible within the training set.

Should any images contain persons or information that we do not have the rights to, these images should be discarded and excluded from the data set.

Developer's expertise

X-Soft is an industry leader in the development of self-driving golf carts. It's CEO Warren Harper, has published multiple academic papers on the development of convolutional neural networks and how they can be used to accomplish deep learning within the computer vision space.

We believe that these credentials make us a suitable candidate for the proposed solution of building a self-driving vehicle for ZCC.

Task 2 - Section B

TECHNICAL PROPOSAL

Problem Statement

Zee Car Company (ZCC) is looking for a way to deliver parts to various locations on its campus and factory floor. The factory runs 24 hours a day and so ZCC needs a solution that can operate continuously.

The solution must not require significant infrastructure change, very minimal human intervention and must be easy to change or adapt to its other locations globally.

Customer summary

Zee Car Company (ZCC) is an automobile manufacturing company. It currently operates eleven factories spread across six different locations. The proposed solution will be utilized specifically by the parts delivery department, however almost all other departments on the factory floor will interact with the proposed solution when receiving deliveries.

Employees will be given a short demonstration in order to familiarize themselves with the vehicle, however it will be designed in such a way that no special training is needed in order to receive deliveries.

There will be one user who is briefed on routine checks that need to be carried out, however we will be able to run diagnostics and push updates remotely.

The autonomous delivery vehicle will operate exclusively indoors and use painted lanes in order to navigate the factories it's operating in.

Existing system analysis

The system currently in place for delivering parts is a manual hand delivery system. ZCC relies on a team of 18 staff that are on a 24 hour schedule to hand deliver

parts to various departments on the factory floor. The initial goal of the proposed solution is to cut the team of 18 down to 3 immediately which will mean that there will be at least one staff member supervising deliveries at all times. Once the pilot is complete there is the potential to deploy more autonomous delivery vehicles while still maintaining the head count of 3.

Data

All training data will be created and collected by X-Soft on behalf of ZCC. There are two distinct data sets that are needed in order to train the two machine learning models that will be utilized in the self-driving vehicle.

1. **Lane navigation data -** This will be a series of images collected by our prototype car. The images will be converted to YUV color space and cropped to a resolution of 200 x 66 which is the optimum resolution and format for our machine learning model. Although a portion of these images will be shot on the ZCC factory floor, there will be no proprietary or sensitive information captured in these images. These images will also not live on the proposed solution. They will be securely stored at X-Soft and then discarded after the model has been trained.
2. **Object detection data -** This will be a collection of images representing the unique objects that the proposed solution will need to detect. The machine learning model being utilized already has common object detection trained into it and so we will only need to train the model on objects unique to the ZCC factory environment. These images will be shot at X-Soft headquarters and will not contain any proprietary or sensitive information. They will not be stored on the proposed solution.

Project methodology

Development will follow the SEMMA methodology,

• **Sample:** Selecting the sample data set. X-Soft will gather images of the factory and potential routes the vehicle might take. X-Soft will also gather images of objects the proposed solution will need to detect.

• **Explore:** X-Soft will try to understand the data (images) and try to anticipate what our program might use in order to anticipate the path of the self-driving vehicle, for example a lane or existing footpath.

• **Modify:** Once X-Soft determines the variables in the explore phase, we'll prepare or modify the data in order to start the modeling process. This could include eliminating unnecessary visual elements from the images, cropping, color space conversion and masking.

• **Model:** We apply data models to the dataset in order to create a certain behaviour from our program. In this case it would be to control the car, make it navigate a lane autonomously, and make it detect predetermined objects

• **Assess:** Evaluate the results of the modeling phase and determine whether or not the desired outcomes were achieved.

Project outcomes

The ultimate project goal is an autonomous self-driving delivery vehicle that can be broken down as follows:

- **Physical vehicle:** design and construction of the physical vehicle. This vehicle will be loosely based on a golf cart design, however, certain modifications will be made to the chassis in order to facilitate parts storage and delivery. The vehicle will be battery powered and a custom charging station will be provided requiring a 3 phase (industrial) electrical outlet for fast charging.

- **On board computer:** The hardware that will run the software to control the vehicle and process its surroundings via a camera in real time. This hardware has been tried and tested in this particular application and will have sufficient power to run the machine learning models.
- **Software:** The main program that will control the vehicle and utilize the machine learning models in order to perform its primary task.
- **Machine learning models:** 2 machine learning models trained specifically for this application and environment that will perform the tasks of lane navigation and object detection.
- **Security system:** The vehicle will be secured using an ID card system. Once the ID card is swiped, the user will have direct access to the system where they will be able to start the application.

Implementation plan

- **Strategy for implementation:** We will implement the proposed solution in a phased manner which will allow us to have tight control over the outcomes. Note that this phased approach will commence after completion of testing.
- **Phases of roll out:** The first phase of roll out will comprise of the vehicle driving to single destinations within the factory to deliver parts. These deliveries will be manned. In the second phase, the vehicle will visit multiple locations in a single run, this phase will still have a staff member manning the vehicle. Lastly, the vehicle will complete multiple deliveries unmanned.
- **Testing and final distribution:** The first phase of testing will be conducted on X-soft premises in a simulated environment that closely resembles a ZCC factory. The second phase of testing will be conducted in a ZCC factory. The vehicle will not pass any phase of testing unless it has a 100% function rate. If it deviates from a path or fails to detect an object then the

model will be assessed and further model training will be implemented. Once the vehicle is deployed it will be under a pilot program where it will be manned 100% of the time until we are satisfied with the accuracy and reliability of the vehicle in the production environment.

- **Dependencies and milestones:**
    - Physical car build
    - Hardware build and installation
    - Software development and installation
    - Data gathering and analysis
    - Model training
    - Testing
    - Deployment
- **Deliverables**
    - Self-driving car
    - Interface to control car
    - Charging station
    - Staff training

Evaluation plan

The self-driving vehicle will be evaluated on two primary outcomes. Firstly its ability to navigate and stay in a specified lane. Secondly, it's ability to detect specific objects and make decisions on those detections.

Since the vehicle will be operating in a physical environment around persons and costly equipment that could have lasting financial impact on ZCC should any injury or damage occur, we will be operating under a zero tolerance for error model. This means that the vehicle will not be deployed until it passes all tests and is able to accomplish its tasks with a 100% accuracy rate.

Initial testing will be carried out in a controlled environment resembling a ZCC factory. Once these tests are complete, testing will then take place in a production environment. Lastly, once the vehicle is deployed, it will be operating under a pilot program whereby it is closely monitored and manned by X-Soft staff.

Resource and costs

Resource	Description	Cost
Delivery vehicle	Custom built electric vehicle and charging station	$92,000
Hardware	Onboard computer to run the software including camera and radar	$3000
Python	Software	0
Tensorflow	Software	0
OpenCV	Software	0
Development	3 x Software engineers	$40,000
Installation	Engineer	$10,000
Delivery	Vehicle delivery and training	$5,000
	Total	$150,000

The total cost for the project is $150,000. $55,000 of this cost comprises human resource needed for software development, installation and training.

The total cost includes all testing and physical environment setup for training the vehicle. The cost does not include ongoing maintenance, this will be offered in a separate contract if required by ZCC. X-Soft will retain ownership of the software and intellectual property, this cost includes one 10 year license for one vehicle. Additional licenses will be available to purchase at a discounted price of $5,000 per license.

Timeline and milestones

The project shall be completed in 4 sprints, taking a total of 6 weeks to complete.

Sprint	Start	End	Tasks
1	09/01/2021	09/13/2021	Data acquisition Data processing Simulation setup
2	09/14/2021	09/31/2021	Hardware install Model training + optimization Vehicle testing
3	10/01/2021	10/13/2021	Dashboard development Testing of vehicle in the intended environment
4	10/14/2021	10/16/2021	Delivery of vehicle Onsite staff training

Task 2 - Section D

POST IMPLEMENTATION REPORT

Project purpose

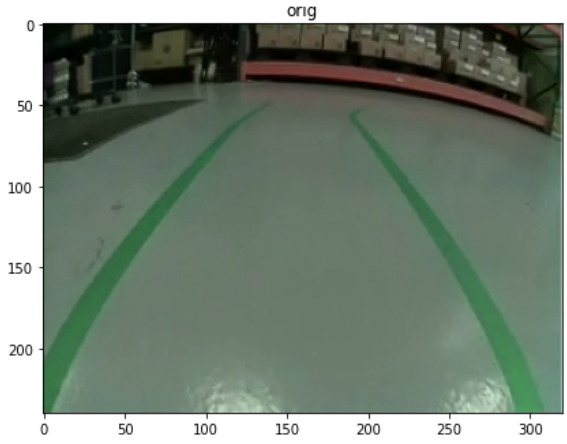
To build a self-driving vehicle that is able to navigate a lane and detect objects using machine learning.

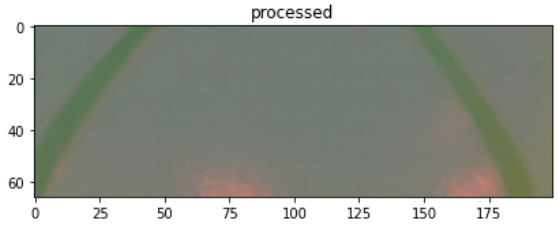
Data sets

Data set 1: Lane navigation

The left image is the raw format and right is the image after preprocessing (processed in the Colab Jupyter notebook ‘arnie_lane_nav.ipynb’). The image is cropped to a resolution of 200 x 66 pixels and converted to the YUV color space. This is optimum size and format for the training the chosen model.

There are 4157 images found in /data/driving_images

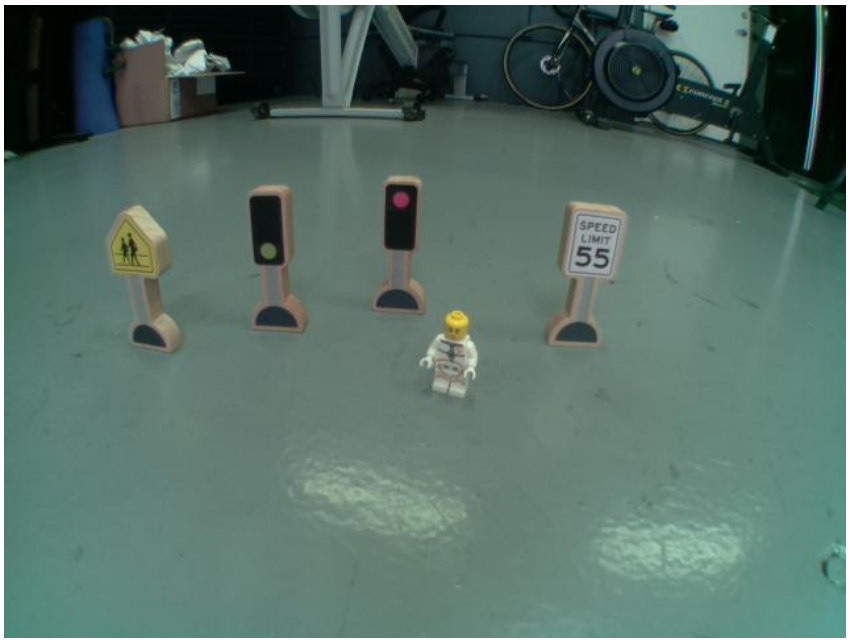


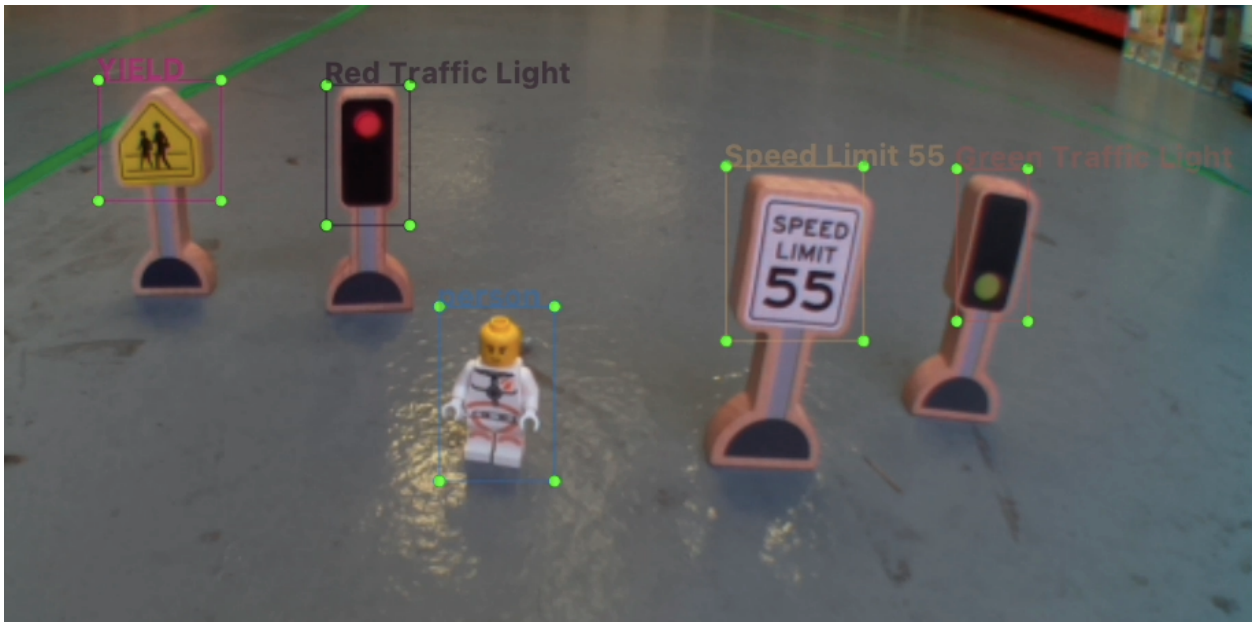


Data set 2: Image detection

These images are used to train the second model to detect these specific objects. The first image is a raw image. The second image is a screen shot of the labelling process. We use

the labelling process to generate an xml file that we use to tell our model the boundaries of our object when training.





Data product code

The root directory consists of 2 Jupyter notebook files that contain the code to prepare our data and subsequent training of the machine learning models.

arnie_lane_nav.ipynb : Written in python in a Jupyter notebook (Colab).

This program:

Imports the training images

Processes the images and prepares them for training

Trains and creates a machine learning model

arnie_sign_detect.ipynb : Written in python in a Jupyter notebook (Colab). This program:

Imports the training images

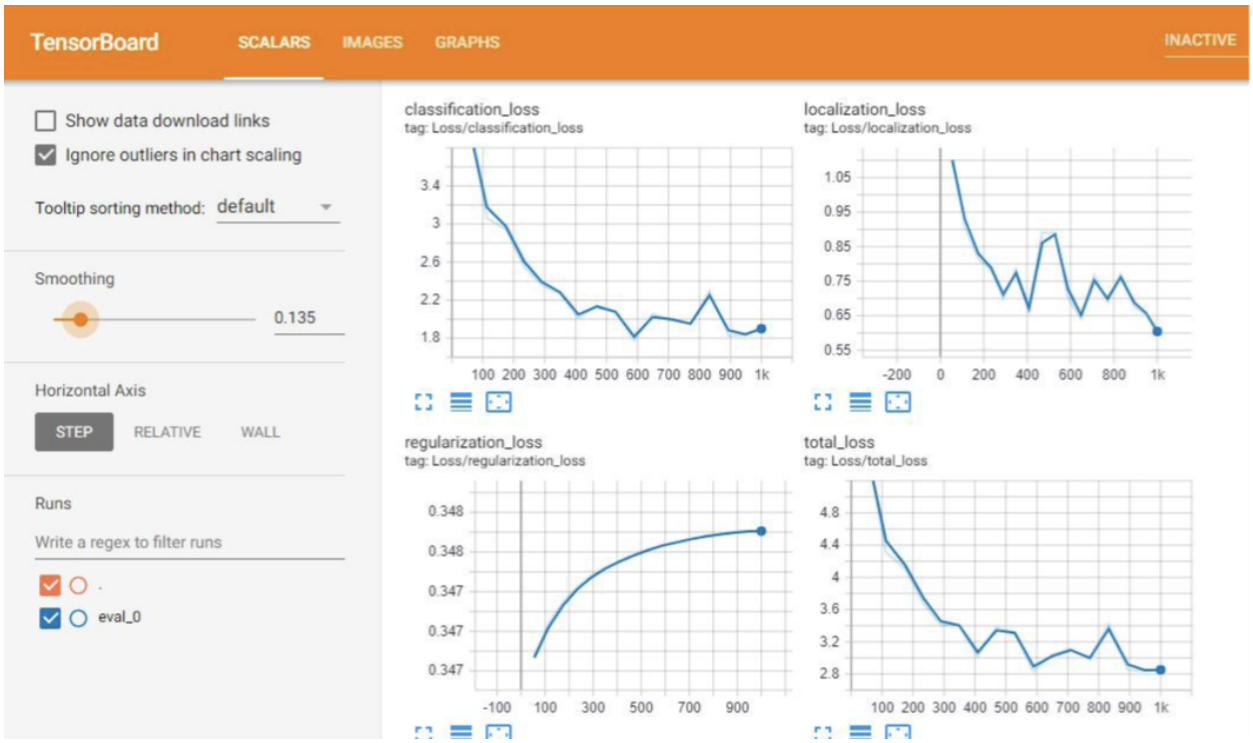
Processes the images and prepares them for training

Trains and creates a quantized (required for TPU) machine learning model

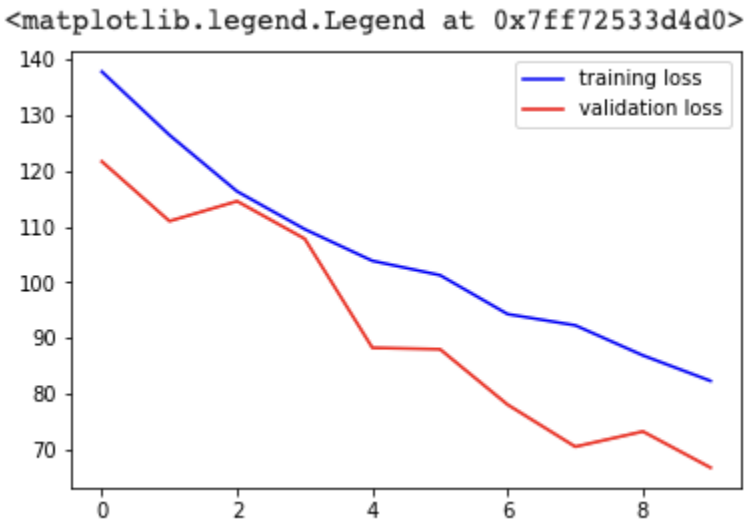
Compiles ML model for TPU use

The trained models can be found in **root/models/**

Below is a Tensorboard screenshot showcasing the classification loss decreasing as the object detection model is being trained.



The second screenshot is a matplotlib graph showcasing the training/validation loss decreasing as the navigation model is trained.



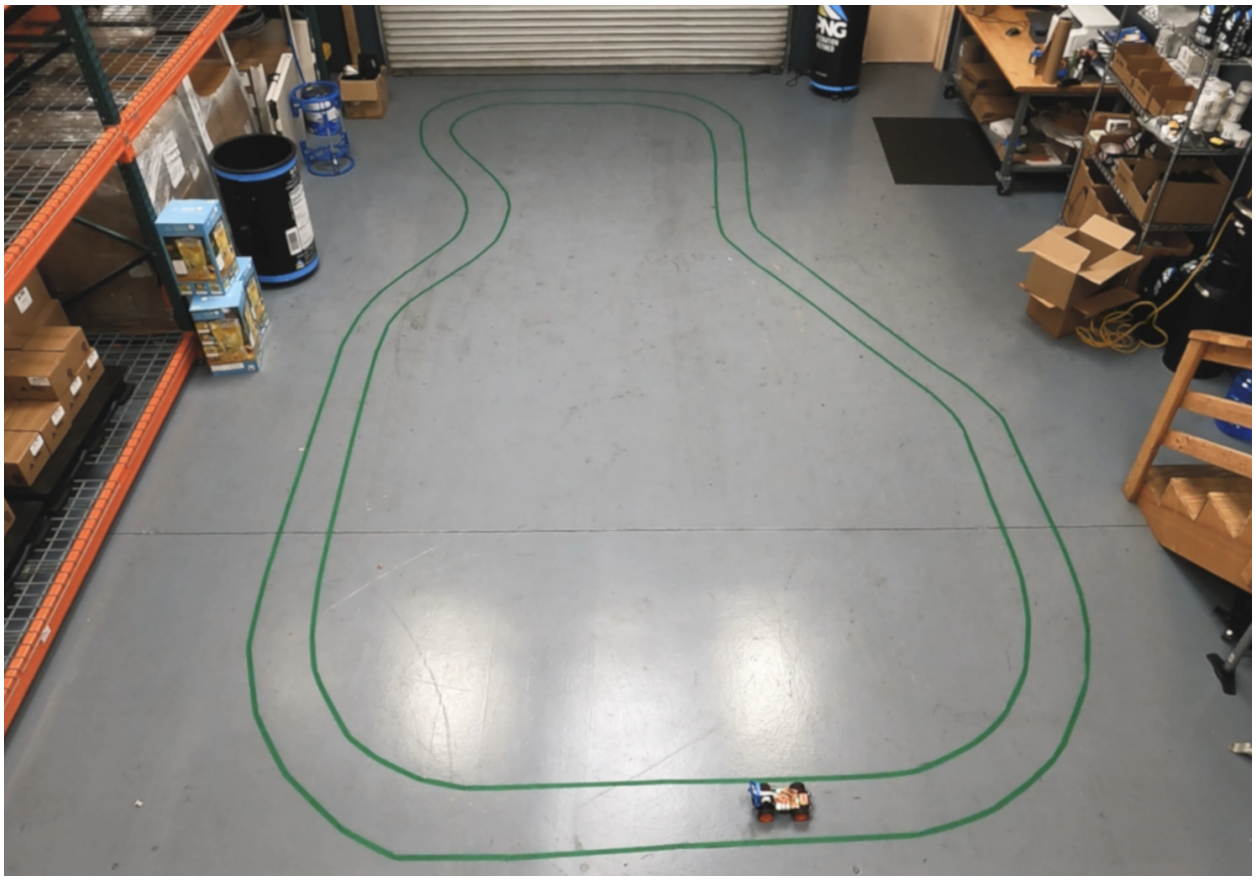
Hypothesis verification

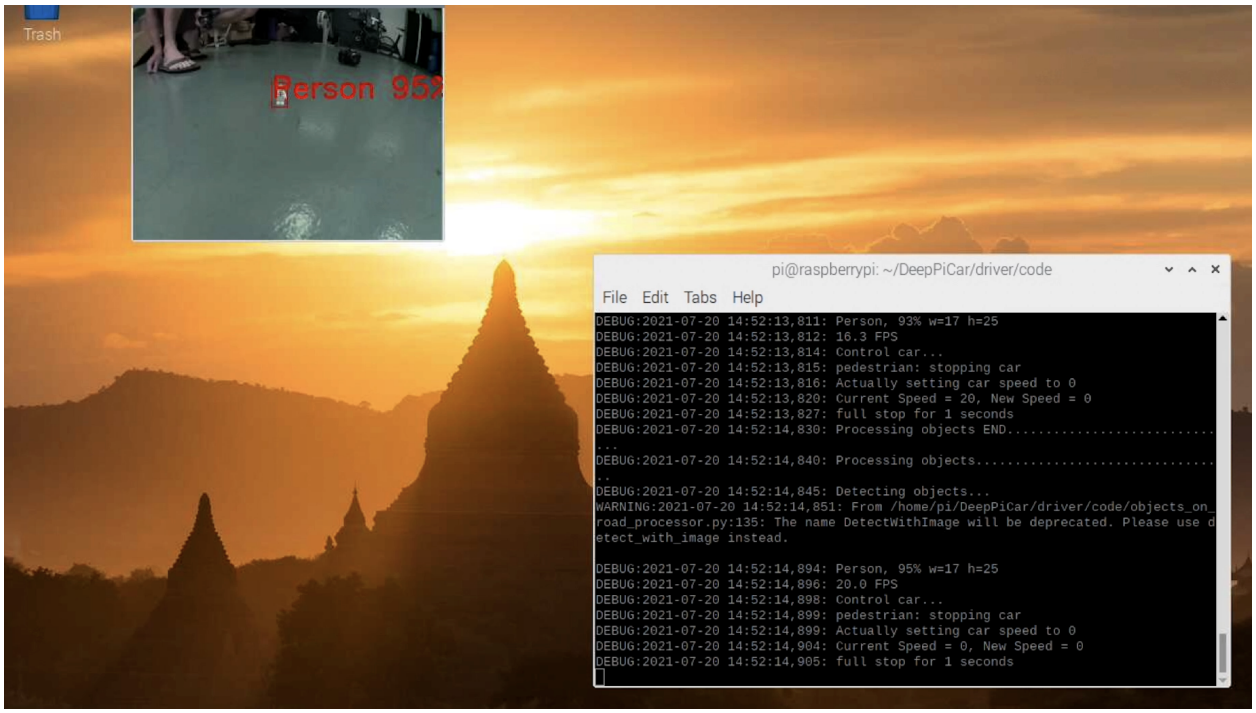
The self-driving car (Arnie) successfully accomplished its two primary goals,

1. Navigate an unseen course.

2. Detect predetermined objects and react accordingly.

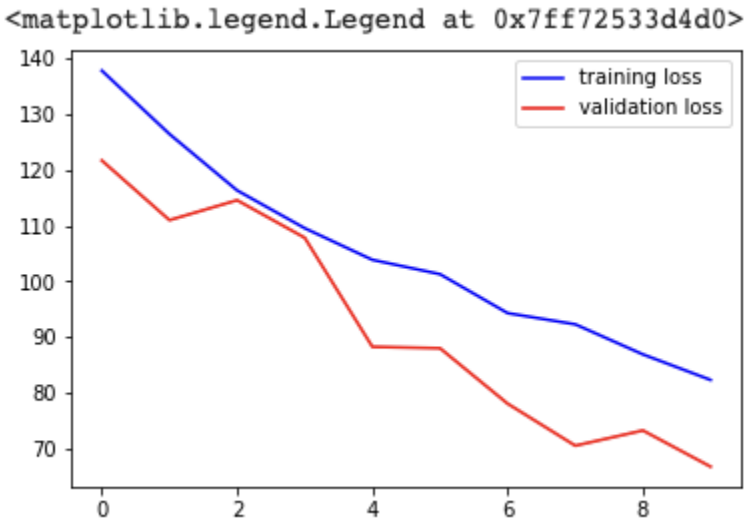
Please see my video which I have included in my capstone submission. Below are two screenshots showing the unseen course and object detection.





Effective visualization and reporting

Below is a graph of the model training process which shows both the training and validation loss converging and lowering, proving that the accuracy of the model is increasing with each training cycle.

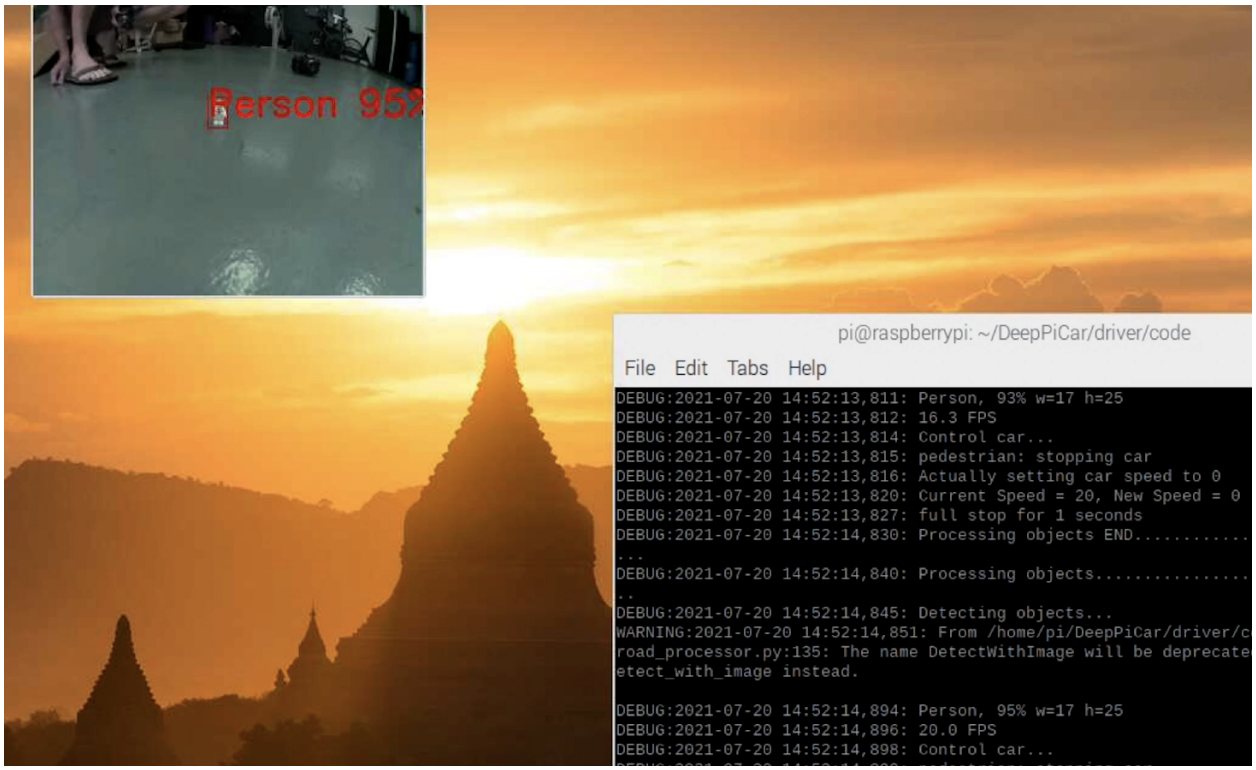


Accuracy analysis

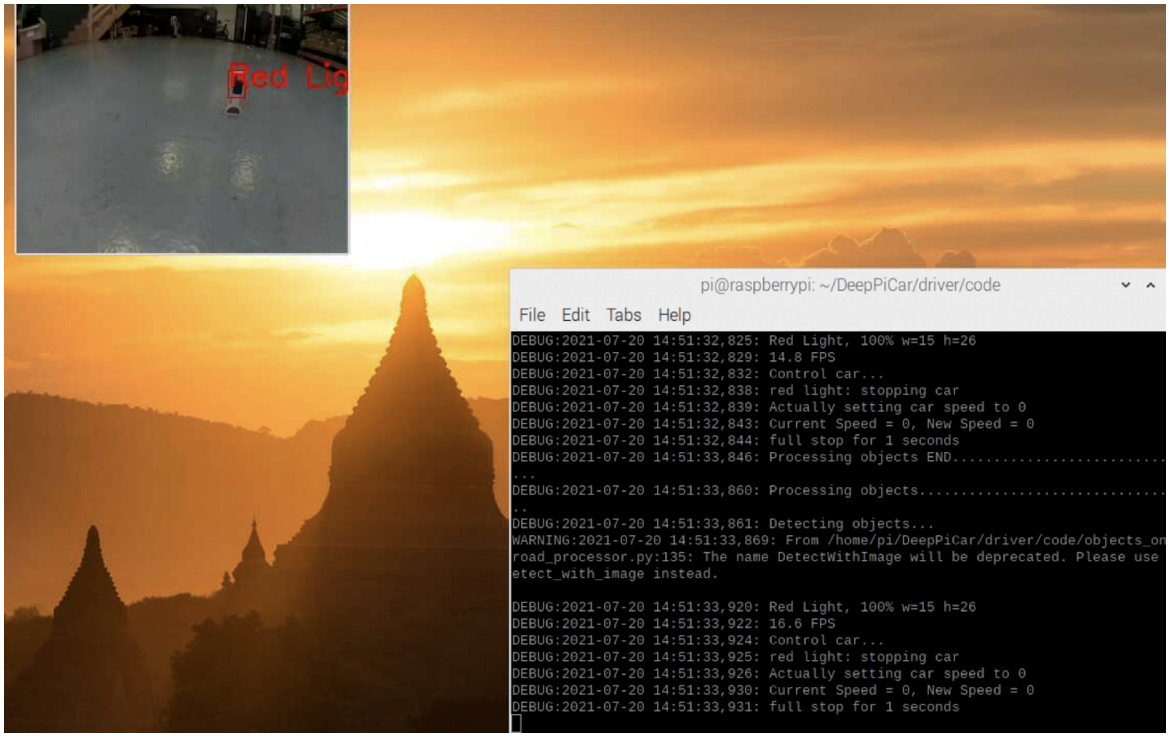
The accuracy of the lane following function was very high (see video). The only time the car would start to diverge is when the speed was set too high, however, I believe this was more a limitation of the hardware (camera and raspberry pi) and its inability to process the data quick enough.

The object detection model was also highly accurate, successfully detecting each object with an assurance of 90% or above.

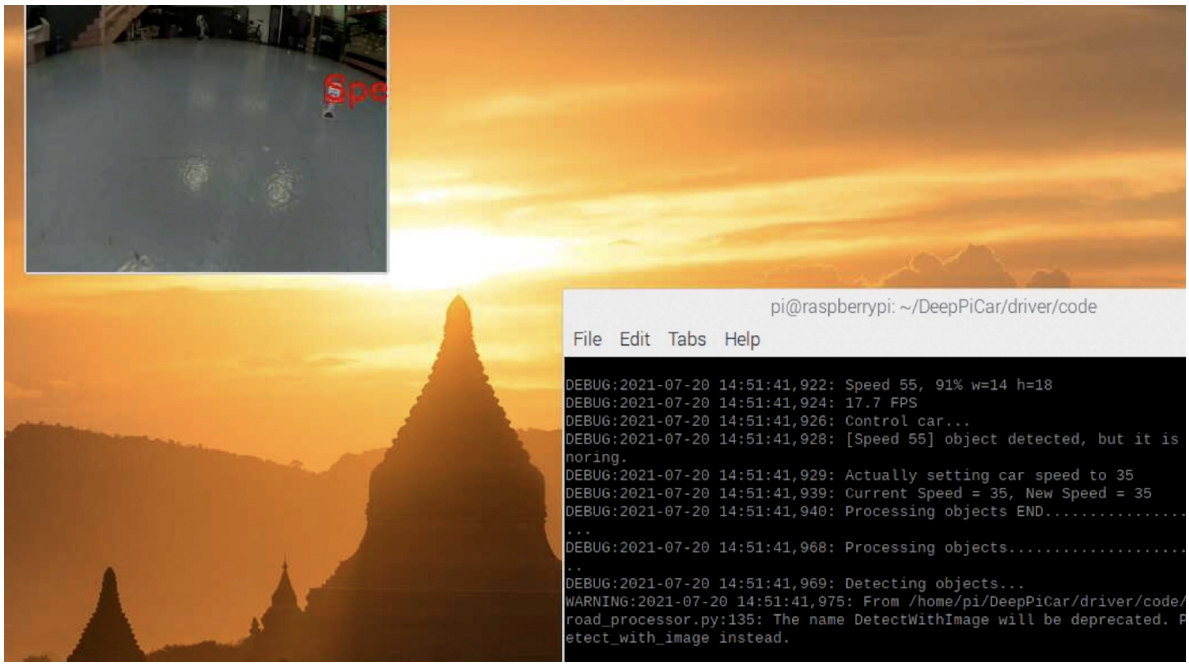
Person - 95%



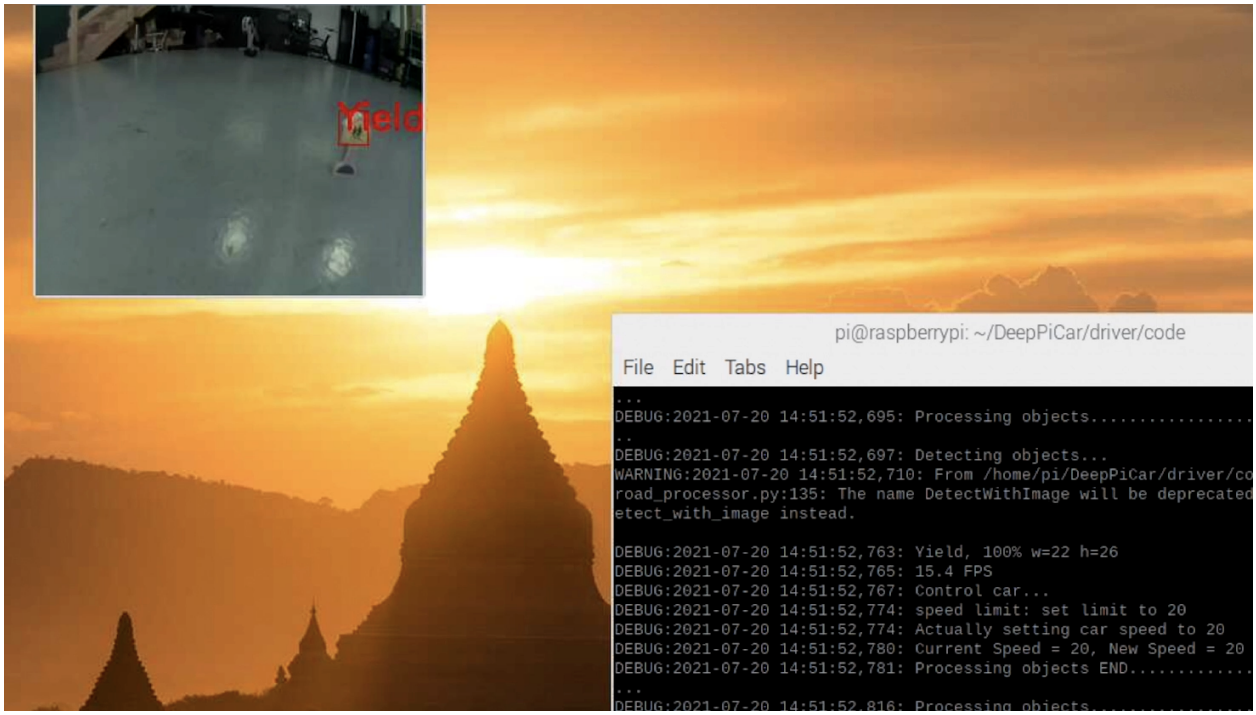
Red light - 100%



Speed sign - 91%



Yield sign - 100%



Application testing

Testing was focused on the two main objectives, lane detection and object detection. The lane navigation was the most challenging. I discovered early on that there are many factors that really influence the models ability to detect a lane successfully. A good example of this is light. When the light changes, the incoming images change. Controlling for the light helped me to increase the accuracy of the lane detection model without providing training images that account for all manner of light situations. In a real world production situation, we would simply collect training data (images) of lanes in as many light/environmental situations as possible.

Object detection was much more straightforward. The model was able to detect the objects very accurately (90% and above) with only 50 training images. It was able to do this in most environments, much less temperamental than the lane navigation model. The only challenge was getting the model to run quick enough on the raspberry pi. During the

initial tests, the raspberry pi cpu could only run the program at 1-2 frames per second. The solution ended up being to implement a dedicated TPU. This was able to run the model at around 15 frames per second.

Due to the nature of the project, no user testing was conducted.

Application files

The root directory consists of 9 code files:

arnie_controller.py : Written in python to control the car manually and record a series of images and the corresponding steering angle of the car. Executed from the command line, the car is then controlled using 'WASD' keys and the program is quit using the 'q' key. These images are used to train the machine learning model.

arnie_lane_nav.ipynb : Written in python in a Jupyter notebook (Colab).

This program:

Imports the training images

Processes the images and prepares them for training

Trains and creates a machine learning model

arnie_sign_detect.ipynb : Written in python in a Jupyter notebook (Colab). This program:

Imports the training images

Processes the images and prepares them for training

Trains and creates a quantized (required for TPU) machine learning model

Compiles ML model for TPU use

arnie.py : Written in python to drive the car, executed from the command line. This program takes one argument in the form of a number from 1-3 to select the feature to use.

1 - Lane detection using OpenCV and math (no machine learning)

2 - Lane detection using machine learning. Car will start driving and navigate between the lines.

3 - Object detection using machine learning. Car will start driving and react to the following objects:

1. Person (lego man) - Car will stop
2. Green traffic light - Car will continue
3. Red traffic light - Car will stop
4. 55 Speed limit sign - Car will speed up
5. Yield sign - car will slow down

detectable_objects.py : Written in python. This is a class utilized by arnie_sign_detect.py to classify the 5 objects listed above.

export_image_steering_data.py : Written in python and executed on the command line. This program takes a video file as an argument and saves it out as a series of images and the corresponding steering angle, calculated using the lane_follow.py program. The images are used to train the lane-following machine learning model.

lane_follow.py : Written in python. This is a class that the main driver program (arnie.py) uses to navigate a lane using edge detection (OpenCV) through the onboard camera and then calculating a heading based on those edges.

lane_follow_ml.py : Written in python. This is a class used by the main driver program (arnie.py) to navigate a lane using machine learning.

object_detection_processor.py : Written in python. This is a class used by the main driver program (arnie.py) to detect objects. This program also provides the logic (what the car needs to do) for each object.

Users guide

The entire code directory including the models need to be copied to a compatible car which utilizes a raspberry pi running NOOBS Raspbian Operating System. This car needs to be a Sunfounder PiCar as the programs use the Sunfounder API in order to control the hardware.

The car needs to have Python 3.x and Tensorflow 2.0 installed. Note that later versions of Tensorflow will not work on the raspberry pi. The car will also need a Coral TPU (via USB) installed in order to run the object detection model.

To run the car, execute arnie.py from the command line and give it one argument to specify the mode which you would like the car to run in.

Example: 'python arnie.py 2' will run ml mode:

mode 1: manual_lane_follow uses the hand coded lane follower (non ML)

mode 2: ml_lane_follow uses the trained machine learning model to follow lane

mode 3: object_detect uses the machine learning model to detect objects

The car can technically run more than one mode however the hardware struggles to keep up. For now I have designed it to run only one mode at a time.

Summation of learning experience

When I set out to do this project I really thought I would be some kind of machine learning master by the end. What I learned during this project is how much I don't know about the subject and how much I still need to learn (which is exciting).

What I have learned throughout my degree program and especially with any classes involving programming is that no project goes exactly to plan and that you almost always spend time on things that you never anticipated. For example, I spent a lot of time getting packages (Tensorflow, keras) to work with the hardware. Unfortunately, the latest versions of all these packages don't work with the raspberry pi and so it started a game of cat and mouse where I would get one package to work and then something else (another dependency) would break as it was no longer compatible with the newly installed package.

Getting the object detection model to work with the TPU also proved very challenging. I learned that the model needed to be quantized and then compiled in a certain way for it to work with that particular hardware. This was made even more difficult as the latest compilers all use versions of Tensorflow that are not compatible with the raspberry pi.

The physical nature of the project also proved challenging. Finding and constructing an environment in which the car could successfully learn to drive without spending thousands of hours training the model proved quite difficult.

I have a lot to learn, but this project has really solidified my decision to pursue software engineering as a career. I am an eternal tinkerer that loves to solve problems and I can’t imagine I would ever get bored in this lifetime.