Autonomous Vehicle Research Project

Brendan Flood
University of Dallas

Patrick Lau
City University of Hong Kong

Julian Halloy
Maryville High School

(RECSEM 2019, Joint Institute for Computational Science, University of Tennessee)
(Dated: August 2, 2019)

1. Abstract

The idea of a self-driving car is one which is actively studied and tested for use on the road. In addition, the machine learning tools required to create such a vehicle has become more and more available to the public as time goes on. With a number of different libraries and softwares available for free download to design and train neural networks and with affordable but powerful miniature computers on the market, one can explore the possibility of creating a self-driving vehicle. The goal of our project was to construct such a car on a small scale using parts and software that are accessible to anyone on an affordable budget ($250), and to test the effectiveness of MagmaDNN software neural networks on training such a car.

2. Background

A large portion of machine learning involves neural networks, which are programs which take input data in the form of numbers and apply a number of matrix multiplications and operations to them to designate a certain type, or output class, identifying what sort of thing that data sample is. To do this, at each step of the data transformation, a network will either multiply the values in the current matrix by another matrix containing predesignated values called weights which help distinguish between output classes, apply a certain function to each matrix value, or add a matrix of predesignated numbers called biases. at the end of the network, each output class is assigned a probability value, and that with the highest probability is designated as the one which the network believes to be correct. When training a network, each data sample is given a label designating which output class it actually belongs to, such that if the network guesses wrong, then the weights are changed based on the differences for each output class between the ideal probability value (1 for the correct output class and 0 for the others) and the given output class (known as the loss). In training, the network will look at a set of data inputs of a designated size, known as a batch, and alter the weights after each set. When every available piece of data for training is sent through the network, it is known as an epoch. A typical training consists of a number of epochs, often ten or fifteen. To test the effectiveness, the network is used on a much smaller set of data which was not used to train, merely comparing the outputs to the actual output class answers to determine the network's accuracy rather than changing the network in any way. [1]

In the instance of image classification, which is the function of a neural network which controls a self-driving car, convolutional networks are the standard and most effective mechanism. A convolutional network is a neural network which looks at a window of pixels from an input image and assigns a matrix value based on all the values of the pixels rather than one individual one, and this window is moved along the entire image, sometimes with overlap between successive windows, to process the input.

3. Objective

This project had two main objectives. The first was to construct a miniature car which can drive itself around the second floor of the part of the Claxton Complex below the JICS. The second objective was to demonstrate the capability of MagmaDNN software to control such a device.

4. Equipment list

This section lists all components we used to build the autonomous vehicle. Some of the parts are 3D printed, and we provide the .STL files in the appendix to print these parts.

Common parts

Parts	Quantity	Notes
Nvidia Jetson Nano	1	
SanDisk 64GB Micro SD card	1	
Gens ace 11.1V 2200mAh 3S 25C LiPo Battery Pack with XT60 Plug	1	
DROK Buck Converter Module	1	Adjust to 5V 4A output
DC Male Power Cable Connector Plug Adapter	1	Solder to the output side of VRM
Car Frame	1	3D Printing part
Arduino Case	1	3D Printing part
Jetson Nano Case	1	3D Printing part
USB 2.0 Type A Male to Type B Male Power Data Transmission Cable	1	30cm, for the Arduino UNO board
XT60 Male Connector	1	Solder to the input side of VRM

Camera

Parts	Quantity	Notes
Raspberry Pi Camera Module V2	1	
Camera Case	1	3D Printing part
Camera Arm	1	3D Printing part

Wi-Fi

Parts	Quantity	Notes
Panda PAUO5 Wi-Fi Dongle	1	

Assembly Hardware

Parts	Quantity	Notes
M3 Screw	6	12mm long, for car frame
M3 Screw	2	20mm long, for the camera arm
M3 Copper Pillar	6	40mm long
2 Pin Header Jumper Shunt Shorting Block	1	For Jetson Nano Pin J48, DC barrel jack power supply mode
Heat Shrink Tube	1	For soldering
Insulating Tape	1	For soldering

Tools needed

- Pliers
- Soldering iron and solder
- Screwdrivers
- Multimeter

5. Procedures

5.1 3D printing

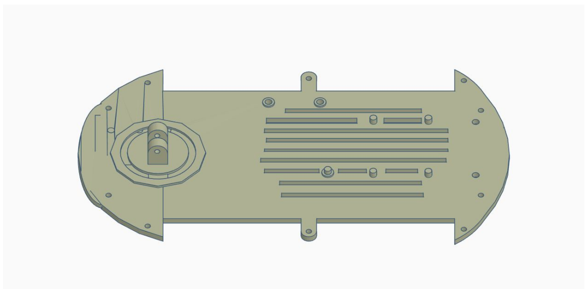
5.1.1 3D printing model design

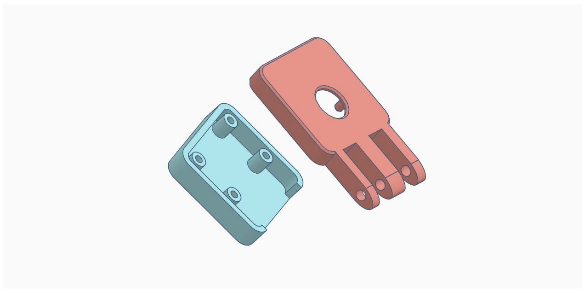
TinkerCAD is used to design the 3D printing models in this project. Since we installed additional components, including camera, Jetson Nano, and power source, on the car unit, we need extra space to put all the individual modules together and form a Jetson Nano Car.

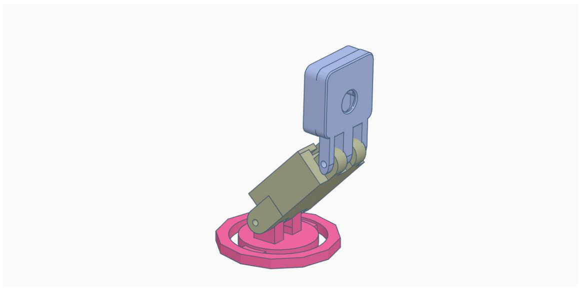
The car frame is placed on the top of the original car by using copper pillars. It provided an extra layer for the car to put the additional parts on there. The pins on the car frame matched with the holes of the base of Jetson Nano Case so that they can be combined without using screw or glue.

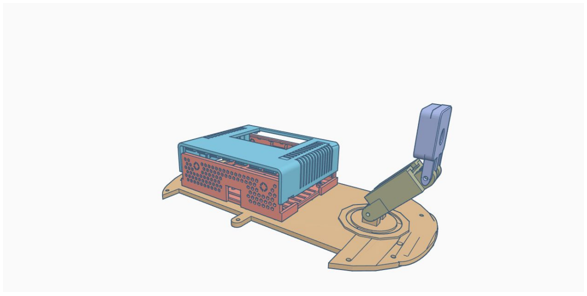
The camera parts are used to protect the Rpi Camera module and allowed the camera to adjust its angle. The arm is only needed to extend the reach if required.

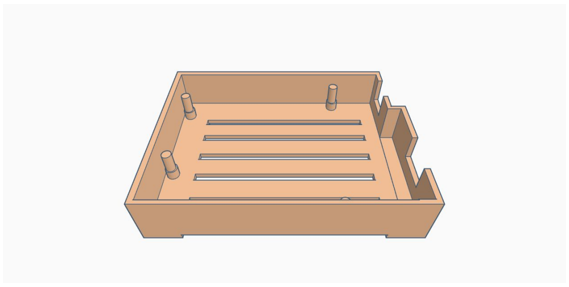
The cases provided simple protection that avoid physical damages when the car crashed into the walls, and also made the car looks cool.

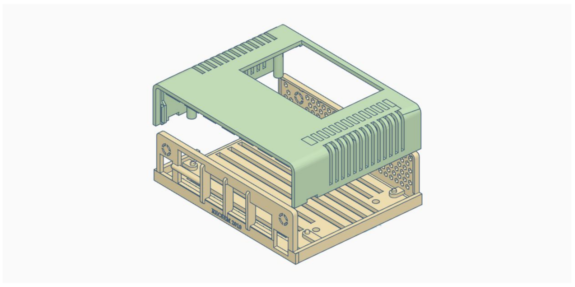

(Fig. 1: Car Frame with Camera Mount)


(Fig. 2: Camera Case)


(Fig. 3: Camera Arm with Mount and Case)


(Fig. 4: Overview of Extra Layer)


(Fig. 5: Arduino Case)


(Fig. 6: Jetson Nano Case)

5.1.2 3D printer setting

Quality

Layer Height	0.15mm

Shell

Wall Thickness	0.8mm
Top/Bottom Thickness	0.6mm

Infill

Infill Density	20%
Infill Pattern	Grid

Speed

Print Speed	60 mm/s
Wall Speed	30 mm/s
Top/Bottom Speed	30 mm/s
Travel Speed	120 mm/s
Initial Layer Speed	30 mm/s

Cooling

Fan Speed	100%
Minimum Layer Time	10 s
Minimum Speed	10 mm/s

Support

Support Overhang Angle	50 degree
Support Horizontal Expansion	0 mm

Build Plate Adhesion

Build Plate Adhesion Type	Skirt
Brim Width	5 mm

5.2 Hardware Setup

5.2.1 Assemble the Car

In this project, Elegoo Car Kit V2 is used, the construction of the car is simple, just follow the official instructions that came with the car.

5.2.2 Camera & Installation

The Jetson Nano Developer Kit supports Mobile Industry Processor Interface Camera Serial Interface (MIPI-CSI) cameras. MIPI-CSI protocol supports high-speed data transmission between cameras and host devices as it is directly connected to the processor, there is not much overhead compared with USB cameras. Therefore, it is suitable for our autonomous vehicle which requires real-time images captured for movement predictions.

Raspberry Pi Camera Module V2

In our project, the Raspberry Pi (RPi) Camera Module V2 with ribbon connector is used. It connects to the board using a simple connector. Besides, the Jetson Nano Developer Kit has an RPi camera compatible connector and device drivers for the Sony IMX 219 imager are pre-installed. So, the camera is configured, and simply plug it in the Jetson Nano J13 Camera Connector.

Testing

Once installed the camera module, it would show up on /dev/video0. Then, test the camera with GStreamer, which is used to interface with cameras on the Jetson Nano. The command line below is used to test the camera:

```
$ gst-launch-1.0 nvarguscamerasrc ! 'video/x-raw(memory:NVMM),width=1920, height=1080,
framerate=30/1, format=NV12' ! nvvidconv flip-method=0 ! 'video/x-raw,width=640, height=480' !
nvvidconv ! nvegltransform ! nveglglessink -e
```

The command requests GStreamer to create a camera stream with 1920x1080 at 30 frames per second and display it in a window with a size of 640x480.

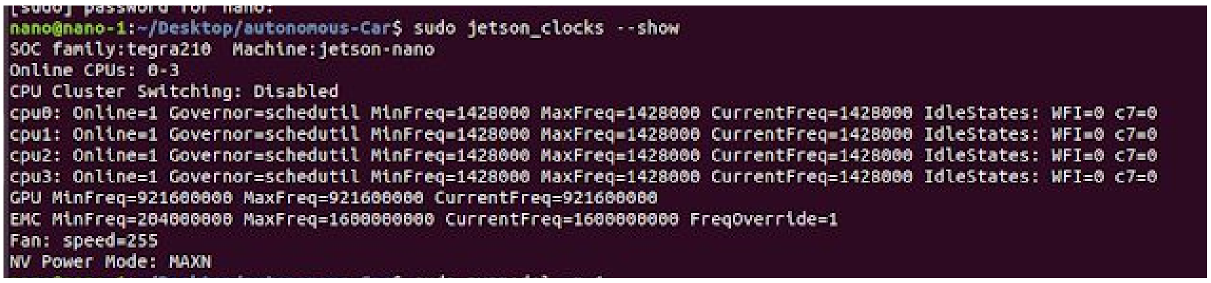
"nvvidconv filp-method" is useful when the mounting of the camera is of a different orientation than the default.

Parameter (nvvidconv flip-method = X)	**Mode**
0	Default, no rotation
1	Rotate 90 degrees counterclockwise
2	Rotate 180 degrees
3	Rotate 90 degrees clockwise
4	Flip horizontally
5	Flip across upper right and lower left diagonal
6	Flip vertically
7	Flip across upper left and lower right diagonal

5.2.3 Power Mode

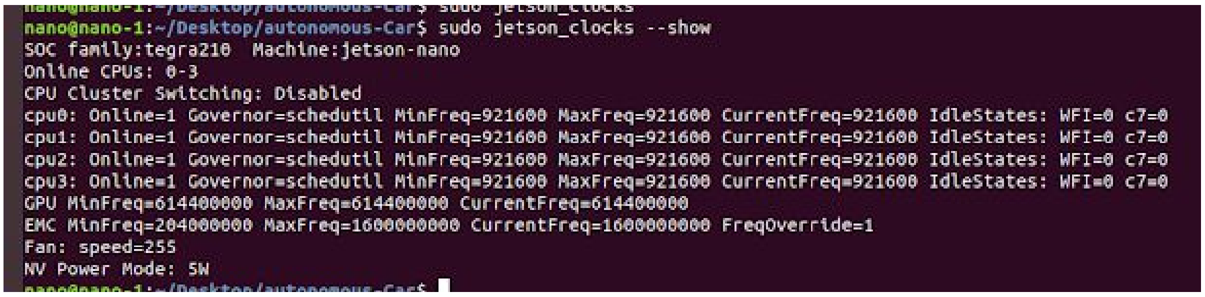
There are two modes for Jetson Nano:

Mode	**Power Consumption**
0	5 W
1	10 W



```
[sudo] password for nano:
nano@nano-1:~/Desktop/autonomous-Car$ sudo jetson_clocks --show
SOC family:tegra210  Machine:jetson-nano
Online CPUs: 0-3
CPU Cluster Switching: Disabled
cpu0: Online=1 Governor=schedutil MinFreq=1428000 MaxFreq=1428000 CurrentFreq=1428000 IdleStates: WFI=0 c7=0
cpu1: Online=1 Governor=schedutil MinFreq=1428000 MaxFreq=1428000 CurrentFreq=1428000 IdleStates: WFI=0 c7=0
cpu2: Online=1 Governor=schedutil MinFreq=1428000 MaxFreq=1428000 CurrentFreq=1428000 IdleStates: WFI=0 c7=0
cpu3: Online=1 Governor=schedutil MinFreq=1428000 MaxFreq=1428000 CurrentFreq=1428000 IdleStates: WFI=0 c7=0
GPU MinFreq=921600000 MaxFreq=921600000 CurrentFreq=921600000
EMC MinFreq=204000000 MaxFreq=1600000000 CurrentFreq=1600000000 FreqOverride=1
Fan: speed=255
NV Power Mode: MAXN
```

(Fig.7: Clock Speed in Mode 0)



```
nano@nano-1:~/Desktop/autonomous-Car$ sudo jetson_clocks
nano@nano-1:~/Desktop/autonomous-Car$ sudo jetson_clocks --show
SOC family:tegra210  Machine:jetson-nano
Online CPUs: 0-3
CPU Cluster Switching: Disabled
cpu0: Online=1 Governor=schedutil MinFreq=921600 MaxFreq=921600 CurrentFreq=921600 IdleStates: WFI=0 c7=0
cpu1: Online=1 Governor=schedutil MinFreq=921600 MaxFreq=921600 CurrentFreq=921600 IdleStates: WFI=0 c7=0
cpu2: Online=1 Governor=schedutil MinFreq=921600 MaxFreq=921600 CurrentFreq=921600 IdleStates: WFI=0 c7=0
cpu3: Online=1 Governor=schedutil MinFreq=921600 MaxFreq=921600 CurrentFreq=921600 IdleStates: WFI=0 c7=0
GPU MinFreq=614400000 MaxFreq=614400000 CurrentFreq=614400000
EMC MinFreq=204000000 MaxFreq=1600000000 CurrentFreq=1600000000 FreqOverride=1
Fan: speed=255
NV Power Mode: 5W
nano@nano-1:~/Desktop/autonomous-Car$
```

(Fig. 8: Clock Speed in Mode 1)

To switch the power mode, type:

$ sudo nvpmodel -m 0 Or $ sudo nvpmodel -m 1

for either mode 0 or mode 1.

To ensure Jetson Nano works in the best performance, we switched it into 10W mode.

5.2.4 Power Supply

There are three different ways to give power into the Jetson Nano:

Port	Max. Voltage / Current
Micro-USB Connector	5V 2.5A
Barrel Jack Connector	5V 4A
GPIO Header	5V 6A

Power Connector

We ran the Jetson Nano in 10W mode, so it needs 5V 2A to perform normally. However, this value is for the Jetson Nano compute module only, not the whole board and the extra attached peripherals, such as camera, Arduino board, and Wi-Fi dongle. Therefore, if we use Micro-USB cable to supply current, the Jetson Nano car would turn itself off randomly due to overdraw of current.

In this project, a barrel jack connector is used to connect the 5V 4A power supply and the Jetson Nano. To enable the barrel jack connector, a jumper is needed to insert on pin J48 of Jetson Nano. Pin J48 is located between the camera connector and the barrel jack connector. It tells the

Jetson Nano circuit to use the barrel jack instead of the micro-USB port to supply power, but the micro-USB port is still available for data transmission with a computer.

Power Source

Supplying a sufficient amount of power to the Jetson Nano was important as the neural networks were process-intensive. The Jetson Nano has two power modes: 5 watts and 10 watts. The 5W mode would work with the micro-USB connector but it would make the Jetson Nano too slow. The 10W mode with the barrel jack connector would be the best for our situation. The lithium-ion (Li-ion) battery pack (5V 2A) did not supply enough current, causing shutdowns while driving. We then used a lithium polymer (LiPo) battery which could supply the necessary current. We used a voltage regulator module (VRM) to ensure the Jetson Nano received the correct voltage and current (5V 4A) from the barrel jack connector.

5.3 Software Setup

5.3.1 Initial Setup

Jetson Nano

The Jetson Nano Developer Kit does not have any eMMC (Embedded Multi-Media Card) or SSD (Solid State Drive) for storage space, so we have to prepare a microSD card. A microSD card of at least 64GB is recommended because we need much space for collecting images. For the first boot, we followed the instructions from Nvidia and created the Linux account on the Jetson Nano.

Basic Setup

After the Jetson Nano boots up, run the script "install_basics.sh" in bitbucket and fix the CUDA toolkit related paths issue. For example, the default path of "nvcc" is not in ${PATH}. The script would add CUDA files into the LD_LIBRARY_PATH and PATH variables. After that, the environment variables would setup properly at the next login.

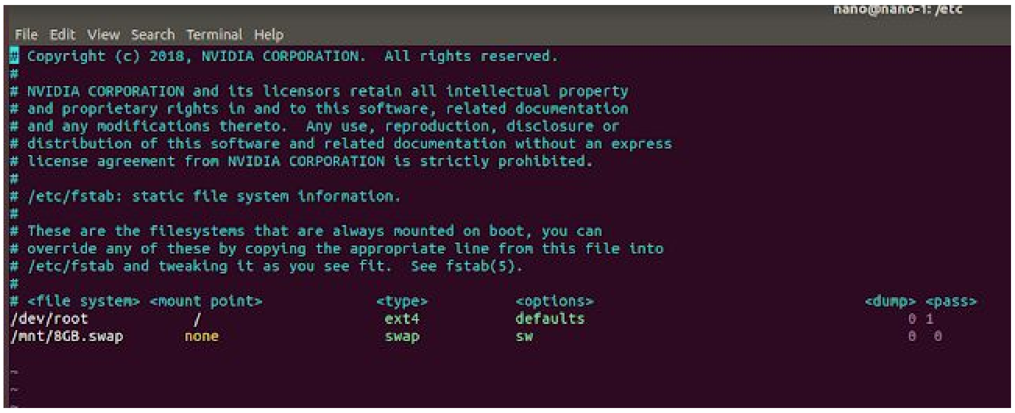
Create Swap File

Then, we created a swapfile on the system and used as virtual memory. Swapfile is a feature in the Linux Kernel which implements paged memory, it is a space on a storage space used as an extension of computer RAM. The Jetson Nano has 4GB of RAM which is quite limited, it would probably run out of memory when it is installing OpenCV, TensorFlow or running autonomous car programs. Therefore, we made an 8GB swapfile by the commands below:

```
$ sudo fallocate -l 8G /mnt/8GB.swap
$ sudo mkswap /mnt/8GB.swap
$ sudo swapon /mnt/8GB.swap
```

Next, we went to /etc/fstab and added the following line:

```
/mnt/8GB.swap          none          swap          sw     0 0
```



```
# Copyright (c) 2018, NVIDIA CORPORATION.  All rights reserved.
#
# NVIDIA CORPORATION and its licensors retain all intellectual property
# and proprietary rights in and to this software, related documentation
# and any modifications thereto.  Any use, reproduction, disclosure or
# distribution of this software and related documentation without an express
# license agreement from NVIDIA CORPORATION is strictly prohibited.
#
# /etc/fstab: static file system information.
#
# These are the filesystems that are always mounted on boot, you can
# override any of these by copying the appropriate line from this file into
# /etc/fstab and tweaking it as you see fit.  See fstab(5).
#
# <file system> <mount point>             <type>         <options>                               <dump> <pass>
/dev/root            /                     ext4           defaults                                   0 1
/mnt/8GB.swap       none                   swap           sw                                         0  0
~
~
```

(Fig. 9: Screenshot of /etc/fstab file)

After reboot, the swapfile would get mounted automatically on the system.

5.3.2 Install OpenCV 3.4.6

To setup the software development environment for our project, we installed OpenCV which is used for capturing images. Although OpenCV 3.3.1 is pre-installed on the Jetson Nano, it does not support GStreamer that cannot utilize hardware H.264/H.265 codec on Jetson Nano. Therefore, we replaced it by OpenCV 3.4.6. To install OpenCV 3.4.6, execute the script “install_opencv-3.4.6.sh” which would download and build the source file from $ ~/src/opencv-3.4.6. The script would take approximately 3 hours for installation. When it finished, OpenCV should able to import (import cv2) in both python and python3.

If timeout error appears, modify the script and add “--default-timeout=100” to following lines.

sudo pip3 install **--default timeout =100** -U protobuf numpy matplotlib
sudo pip2 install **--default timeout =100** -U protobuf numpy matplotlib

If “error: Please include the appropriate gl headers before including cuda_gl_interop.h” appears:

- Goto /usr/local/cuda/include/cuda_gl_interop.h
- Disable line 62 to 68 (except line 67) by comment them
- //#if defined(arm) || defined(aarch64)
- //#ifndef GL_VERSION
- //#error Please include the appropriate gl headers before including cuda_gl_interop.h
- //#endif
- //#else
- #include <gl gl.h="">
- //#endif

Then, remove the opencv-3.4.6 folder (rm -R opencv-3.4.6) in ~/src/ and run the script again.

5.3.3 Build TensorFlow-1.12.2

To build TensorFlow, make sure swapfile is created. Otherwise, it would probably fail due to a lack of memory. At the beginning, we uninstalled the previous version of tensorflow and tensorboard:

sudo pip3 uninstall -y tensorflow tensorboard

Then, we updated libprotobuf (3.6.1) by using the script “install_protobuf-3.6.1.sh” as version 3.6.1 of protobuf is the matching version in tensorflow-1.12 source code. This installation took about 1 hour to finish on Jetson Nano. After that, we install bazel 0.15.2 by using “install_bazel-0.15.2.sh”, it is the building tool for tensorflow. Next, install and build tensorflow by running “install_tensorflow-1.12.2.sh”. It needs approximately 14hours to run. The “pip3 install tensorflow” would update python3 “protobuf” module to the latest version, but we do not need to. Therefore, we have to uninstall the latest version and re-install version 3.6.1 of python “protobuf” module again.

```
$ sudo pip3 uninstall -y protobuf
$ cd ${HOME}/src/protobuf-3.6.1/python
$ sudo python3 setup.py install --cpp_implementation
```

5.3.4 Disable the Ubuntu GUI

We removed the Ubuntu GUI that pre-installed in the Jetson Nano because it consumes a lot of RAM usage in background. The background memory usage decreased significantly after we did that. One important thing is that, make sure you can ssh the Jetson Nano before doing this, because you will not able to use your display as a terminal anymore. Backup before doing this.

Steps:

1. boot to GUI and open a terminal
2. run:

```
$ sudo systemctl enable multi-user.target
$ sudo systemctl set-default multi-user.target
```

3. Reboot and connect using ssh
4. run:

```
$ sudo apt --simulate purge whoopsie
$ sudo apt purge whoopsie
```

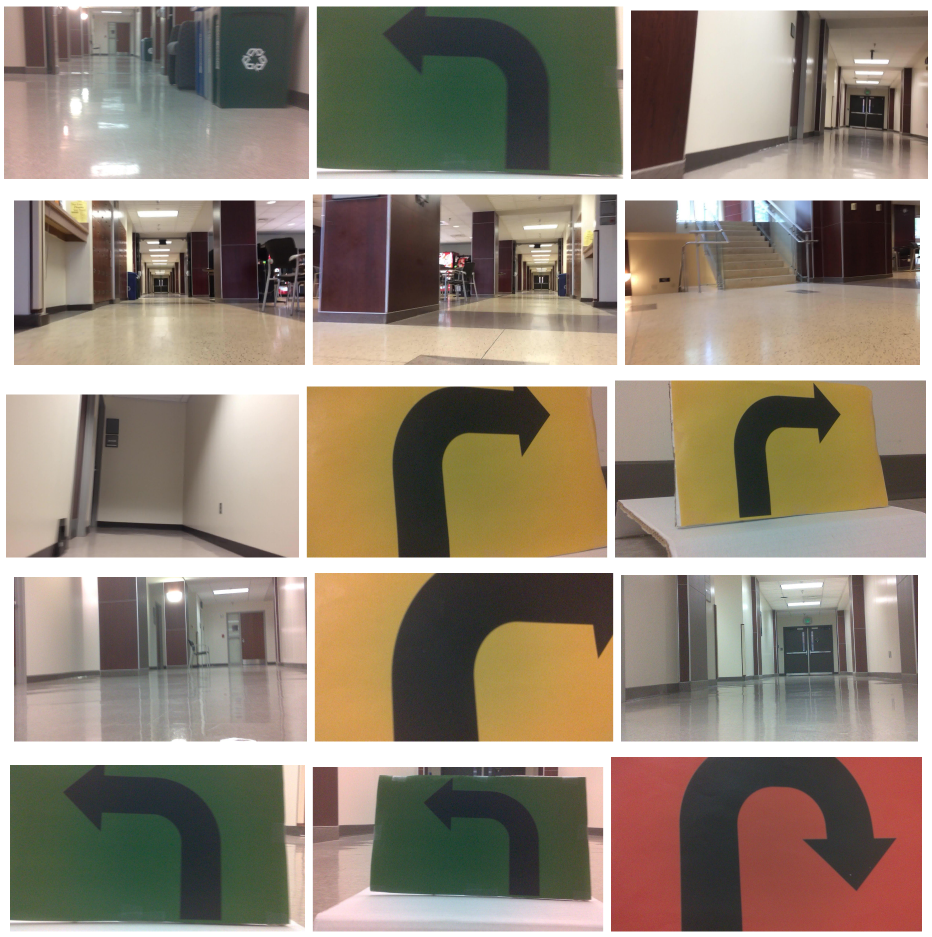
5. Reboot again

5.4 Communicate with Arduino UNO

The Arduino Uno’s USB port can be used not only for sending code, but for serial communication. We used the python library, PySerial, to create a python code that would interface the Jetson Nano with the Arduino. This allowed the Jetson Nano to control the motors of the car, whether that be for manual control or the output of the neural network.

5.5 Data Collection

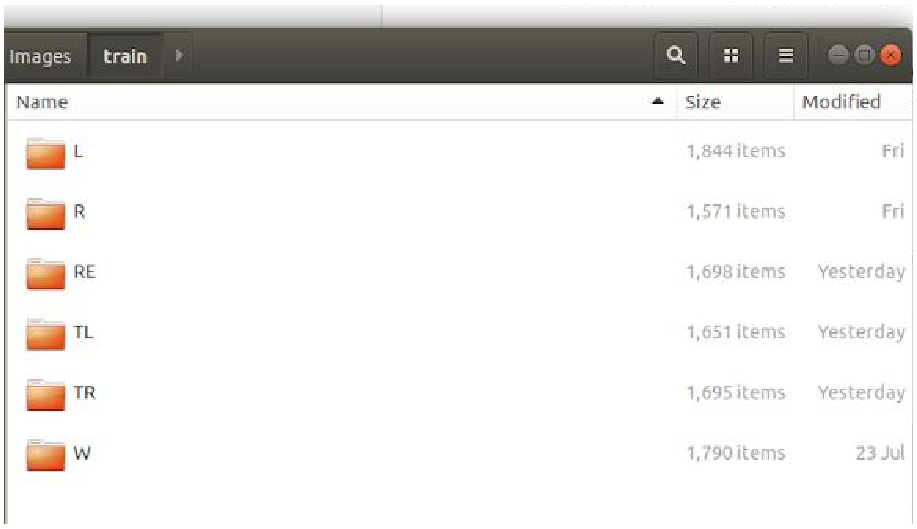
Data Collection was done with the Jetson Nano and the Pi Camera to ensure the most similarity to the images that would be seen by the neural network. Typically, we would use the PySerial code to drive the car manually and run an OpenCV python code that captured images simultaneously.


(Fig. 10: Sample images from collected data)

5.6 Data Labeling

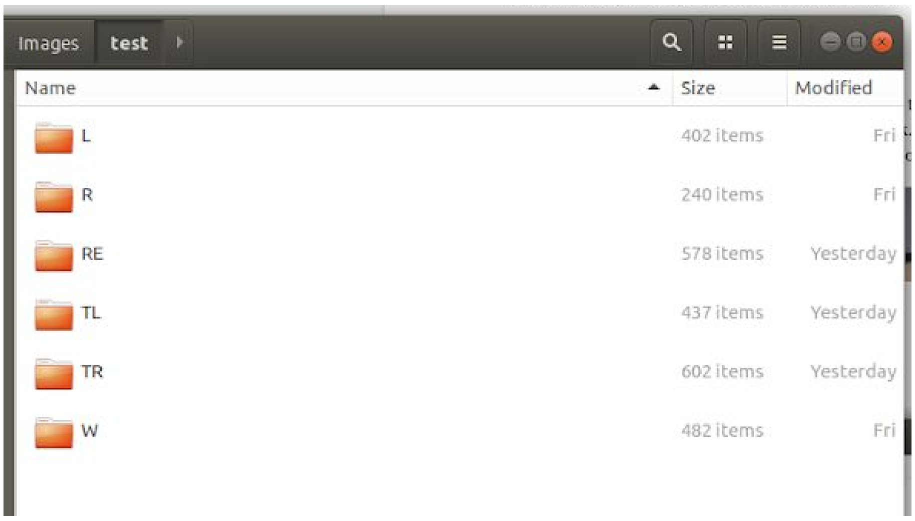
Each image was labeled in such a way that would correspond to the command the Nano would give to the robot in response to it. Earlier on, images were simply labeled "W", "R", or "L". Which meant, respectively, that the car was faced with empty hallway and should go forward, the car was veering toward the wall on the left and should turn right, and that the car was veering toward the wall on the right and should turn left. Once enough images were collected that the car could accurately execute these three commands in response to its environment, traffic signs were introduced giving directives. Left and right turn signs were used at first, but these were insufficient to keep the car from getting stuck in a dead end, so a u-turn sign was introduced. There was some difficulty distinguishing between them since they all consisted of black arrows on small white signs. In the end, left, right, and u-turn signs, each category with a distinct color, were used. This yielded three additional output classes: "TL" in response to a green left turn sign, telling the car to turn 90 degrees to the left; "TR", in response to a yellow right turn sign, telling the car to turn 90 degrees to the right; and "RE", in response to a red u-turn sign, telling the car to turn 180 degrees.

The mechanism by which the data was labeled was that they were sorted into folders corresponding to their output classes after they were collected. Whenever a set of images was collected, they would first be uploaded to Google Drive which was shared among the project collaborators. Next, large groups of images would be selected and moved into new folders named after their output classes (as aforementioned, “W”, “R”, “L”, “TL”, “TR”, and “RE”). one of each of these folders were within the two larger folders “test” and “train”. About four fifths of the images were put into their respective folders within “train” while the rest were placed in that of “test”. Then, both the train and test folders were downloaded onto the workstation. When the training program was run, it would then search the “test” and “train” folders, training the network with those images in “train”, taking the images’ immediate directory names as their labels, and the images in “test” would then be used in a similar fashion to test the network’s accuracy.


Images › train

Name	Size	Modified
L	1,844 items	Fri
R	1,571 items	Fri
RE	1,698 items	Yesterday
TL	1,651 items	Yesterday
TR	1,695 items	Yesterday
W	1,790 items	23 Jul

(Fig. 11: Folders for Model Training)


Images › test

Name	Size	Modified
L	402 items	Fri
R	240 items	Fri
RE	578 items	Yesterday
TL	437 items	Yesterday
TR	602 items	Yesterday
W	482 items	Fri

(Fig. 12: Folders for Model Testing)

5.7 Model Training

The neural network trained to drive the car was the ResNet network from the ImageAI library [2]. The ImageAI library is a free public source for image recognition with pre-trained object locators and trainable neural networks. It runs on Python and required the installation of TensorFlow and Keras, which were rather simple with pip install, and was run in a tensorflow virtual environment. The code for the file which trained the network was remarkably simple, requiring the selection of one of four neural networks, a function to save the model in an h5 file after training, and a training function which took several parameters. These included number of output classes, number of epochs, the option to enhance data (for a low number of data samples), the batch size, and the option to print the network summary. Typically, the model was trained with a batch size of about 20 for a duration of 10 epochs, enhancing the data only when recommended, which was when output classes had less than 1000 samples.

6. MagmaDNN

The implementation of MagmaDNN in this project was limited, due to the amount of time required to give the car self-driving capabilities and difficulties in installing and using it. However, some work was done with it which demonstrates that it can be used on the Jetson Nano.

To give an overview, MagmaDNN is a neural network library which is written in c++. It is designed in a way which is comparable to Keras, the basis for the ImageAI library and a Python neural network library in which network layers are declared in sequence with their parameters and combined with a training function. Although MagmaDNN is optimized for use on GPU and designed to run faster than other neural networks, there were some obstacles and difficulties in attempting to use it.

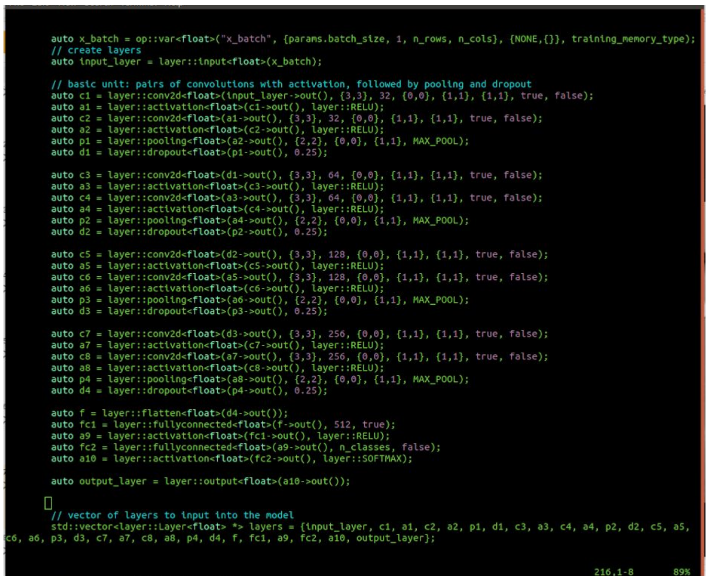
One of these is that there is some difficulty installing it. The source page on github supplies a tutorial for doing so which is rather helpful, but there are a number of dependencies required which may not necessarily be simple to set up properly. These include some form of BLAS (basic linear algebra subsystem), CUDA, CuDNN, and Magma. When installing MagmaDNN, ensuring that each individual dependency refers to the proper library paths of the others can be difficult.

Another issue is the GPU/CPU issue. While MagmaDNN can be used on just the CPU and requires fewer dependencies to do so, on top of the obvious issue that it would run slower, it is incapable of performing some of its critical functions, including convolutional layers. This made the project more difficult since MagmaDNN could not be run on the workstation with its ample memory. While it did have a GPU, it was an older model with insufficient compute power for CuDNN.

Another issue which may not be so much inherent to MagmaDNN as to neural network software in general is that of reading input. The MagmaDNN download comes with a number of examples which were set up to be run on the MNIST dataset. However, for this project and based on how images were labeled for ImageAI, reading the dataset into a network would require extraction of image files from directories and assigning labels based on which directory each image was in. While code was written to do this with the OpenCV library and the c++ recursive file iterator, this code was never tested due to the primacy of giving the car self-driving capability with the ImageAI library.

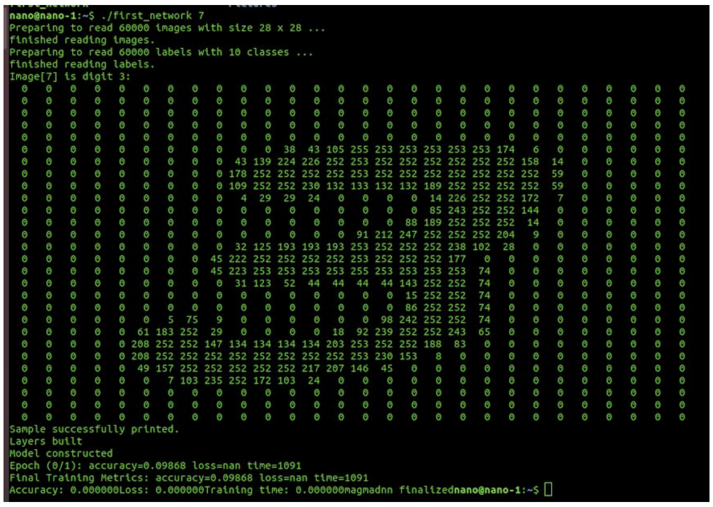
Further difficulties came from the Jetson Nano. One of these was that it came with CuDNN pre-installed, but in an abnormal way that was difficult to find. Thus, in the end, CuDNN was still installed from a download and the original CuDNN was partially removed to prevent conflict. Furthermore, the Nano’s Ubuntu system was set up such that, despite the fact that all the dependencies were installed and the Nano had a GPU, it nonetheless made the CPU version.

Eventually, it came to the point where a medium sized convolutional network could be trained on the nano. A convolutional network designed to eventually drive the car was condensed and trained on the nano. Sadly, with the Nano's limited memory, it could not train particularly fast. The network had 17 hidden layers. It was trained for one epoch on the MNIST dataset, which took about 1000 seconds and yielded an accuracy of 98.6%. This demonstrated that the Nano could be used to run MagmaDNN, with some drawbacks. Since the current version of MagmaDNN does not have a model saving function, any network would have to be trained and run at the same time, which would probably take a long time. This also calls into question whether or not the trained network would run fast enough to identify live images from the Raspberry Pi camera and give an output class soon enough for the car to give a response.



```
    auto x_batch = op::var<float>("x_batch", {params.batch_size, 1, n_rows, n_cols}, {NONE,{}}, training_memory_type);
    // create layers
    auto input_layer = layer::input<float>(x_batch);

    // basic unit: pairs of convolutions with activation, followed by pooling and dropout
    auto c1 = layer::conv2d<float>(input_layer->out(), {3,3}, 32, {0,0}, {1,1}, {1,1}, true, false);
    auto a1 = layer::activation<float>(c1->out(), layer::RELU);
    auto c2 = layer::conv2d<float>(a1->out(), {3,3}, 32, {0,0}, {1,1}, {1,1}, true, false);
    auto a2 = layer::activation<float>(c2->out(), layer::RELU);
    auto p1 = layer::pooling<float>(a2->out(), {2,2}, {0,0}, {1,1}, MAX_POOL);
    auto d1 = layer::dropout<float>(p1->out(), 0.25);

    auto c3 = layer::conv2d<float>(d1->out(), {3,3}, 64, {0,0}, {1,1}, {1,1}, true, false);
    auto a3 = layer::activation<float>(c3->out(), layer::RELU);
    auto c4 = layer::conv2d<float>(a3->out(), {3,3}, 64, {0,0}, {1,1}, {1,1}, true, false);
    auto a4 = layer::activation<float>(c4->out(), layer::RELU);
    auto p2 = layer::pooling<float>(a4->out(), {2,2}, {0,0}, {1,1}, MAX_POOL);
    auto d2 = layer::dropout<float>(p2->out(), 0.25);

    auto c5 = layer::conv2d<float>(d2->out(), {3,3}, 128, {0,0}, {1,1}, {1,1}, true, false);
    auto a5 = layer::activation<float>(c5->out(), layer::RELU);
    auto c6 = layer::conv2d<float>(a5->out(), {3,3}, 128, {0,0}, {1,1}, {1,1}, true, false);
    auto a6 = layer::activation<float>(c6->out(), layer::RELU);
    auto p3 = layer::pooling<float>(a6->out(), {2,2}, {0,0}, {1,1}, MAX_POOL);
    auto d3 = layer::dropout<float>(p3->out(), 0.25);

    auto c7 = layer::conv2d<float>(d3->out(), {3,3}, 256, {0,0}, {1,1}, {1,1}, true, false);
    auto a7 = layer::activation<float>(c7->out(), layer::RELU);
    auto c8 = layer::conv2d<float>(a7->out(), {3,3}, 256, {0,0}, {1,1}, {1,1}, true, false);
    auto a8 = layer::activation<float>(c8->out(), layer::RELU);
    auto p4 = layer::pooling<float>(a8->out(), {2,2}, {0,0}, {1,1}, MAX_POOL);
    auto d4 = layer::dropout<float>(p4->out(), 0.25);

    auto f = layer::flatten<float>(d4->out());
    auto fc1 = layer::fullyconnected<float>(f->out(), 512, true);
    auto a9 = layer::activation<float>(fc1->out(), layer::RELU);
    auto fc2 = layer::fullyconnected<float>(a9->out(), n_classes, false);
    auto a10 = layer::activation<float>(fc2->out(), layer::SOFTMAX);

    auto output_layer = layer::output<float>(a10->out());

   
    // vector of layers to input into the model
    std::vector<layer::Layer<float> *> layers = {input_layer, c1, a1, c2, a2, p1, d1, c3, a3, c4, a4, p2, d2, c5, a5,
c6, a6, p3, d3, c7, a7, c8, a8, p4, d4, f, fc1, a9, fc2, a10, output_layer};

                                                                                              216,1-8       89%
```

(Fig 13: Layer structure of the testing network designed in MagmaDNN)

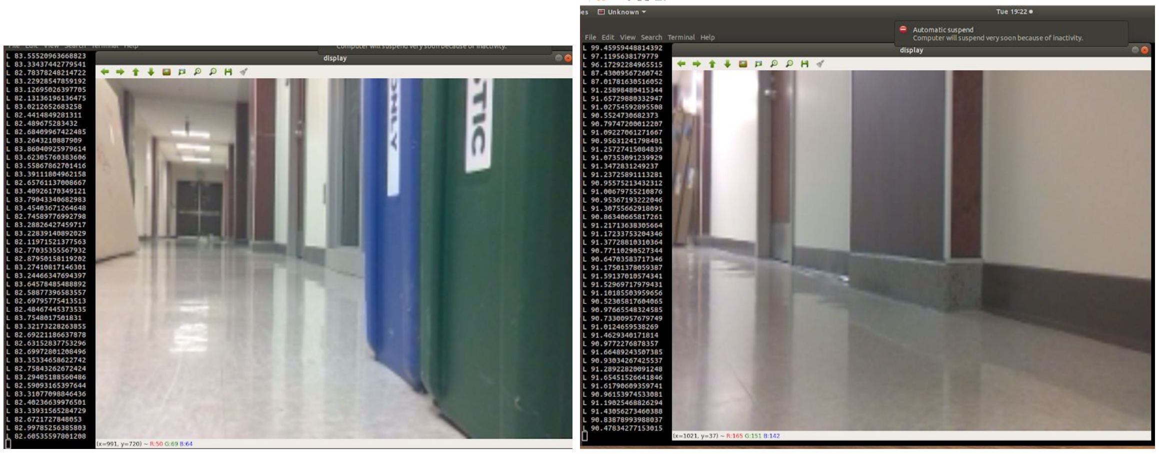


```
nano@nano-1:~$ ./first_network 7
Preparing to read 60000 images with size 28 x 28 ...
finished reading images.
Preparing to read 60000 labels with 10 classes ...
finished reading labels.
Image[7] is digit 3:
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0  38  43 105 255 253 253 253 253 253 174   6   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  43 139 224 226 252 253 252 252 252 252 252 252 158  14   0   0   0   0   0
   0   0   0   0   0   0   0   0   0 178 252 252 252 252 253 252 252 252 252 252 252 252  59   0   0   0   0   0
   0   0   0   0   0   0   0   0   0 109 252 252 230 132 133 132 132 189 252 252 252 252  59   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   4  29  29  24   0   0   0   0  14 226 252 252 172   7   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  85 243 252 252 144   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  88 189 252 252 252  14   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0  91 212 247 252 252 252 204   9   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  32 125 193 193 193 253 252 252 252 238 102  28   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0  45 222 252 252 252 252 253 252 252 252 177   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0  45 223 253 253 253 253 255 253 253 253 253  74   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  31 123  52  44  44  44  44 143 252 252  74   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  15 252 252  74   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  86 252 252  74   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   5  75   9   0   0   0   0   0   0  98 242 252 252  74   0   0   0   0   0   0   0   0
   0   0   0   0   0  61 183 252  29   0   0   0   0  18  92 239 252 252 243  65   0   0   0   0   0   0   0   0
   0   0   0   0   0 208 252 252 147 134 134 134 134 203 253 252 252 188  83   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 208 252 252 252 252 252 252 252 252 253 230 153   8   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0  49 157 252 252 252 252 252 217 207 146  45   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   7 103 235 252 172 103  24   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
Sample successfully printed.
Layers built
Model constructed
Epoch (0/1): accuracy=0.09868 loss=nan time=1091
Final Training Metrics: accuracy=0.09868 loss=nan time=1091
Accuracy: 0.000000Loss: 0.000000Training time: 0.000000magmadnn finalizednano@nano-1:~$ 
```

(Fig. 13: terminal sample from running a designed convolutional network on the MNIST dataset)

7. Results

After we used ImageAI for a while, we had produced several neural networks. The initial networks did not perform well. The car's behavior appeared close to random. After adjusting input images and retraining, the car began to perform better. It could now avoid the walls and drive in a straight line. We continued by adding labels for pictures in which the car should turn left or right. However, the images were too similar to the normal images used to avoid the walls, resulting in an inaccurate network. Our final network utilizes colored arrows to indicate to the car to turn left or right and performs far better.

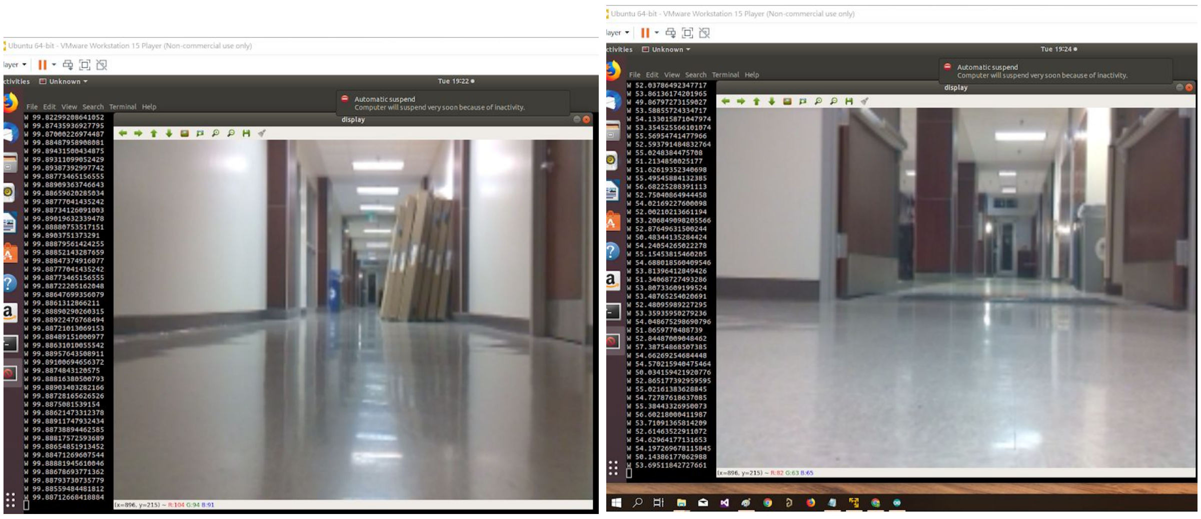


```
L 83.55520963668823
L 83.33437442779541
L 82.78378248214722
L 83.22928547859192
L 83.12695026397705
L 82.13136196136475
L 83.0212652683258
L 82.4414849281311
L 82.489675283432
L 82.68409967422485
L 83.2643210887909
L 83.86040925979614
L 83.62305760383606
L 83.55867862701416
L 83.39111804962158
L 82.65761137008667
L 83.40926170349121
L 83.79043340682983
L 83.45403671264648
L 82.74589776992798
L 83.28826427459717
L 83.22839140892029
L 82.11971521377563
L 82.77035355567932
L 82.87950158119202
L 83.27410817146301
L 83.24466347694397
L 83.64578485488892
L 82.58877396583557
L 82.69795775413513
L 82.48467445373535
L 83.7548017501831
L 83.32173228263855
L 82.69221186637878
L 82.63152837753296
L 82.69972801208496
L 83.35334658622742
L 82.75843262672424
L 83.29405188560486
L 82.59093165397644
L 83.31077098846436
L 82.40236639976501
L 83.33931565284729
L 82.6721727848053
L 82.99785256385803
L 82.60535597801208
```

Automatic suspend
Computer will suspend very soon because of inactivity.

```
L 99.45959448814392
L 97.1195638179779
L 96.17292284965515
L 87.43009567260742
L 87.01781630516052
L 91.25898480415344
L 91.65729880332947
L 91.02754592895508
L 90.5524730682373
L 90.79747200012207
L 91.09227061271667
L 90.95631241798401
L 91.25727415084839
L 91.07353091239929
L 91.3472831249237
L 91.23725891113281
L 90.95575213432312
L 91.00679755210876
L 90.95367193222046
L 91.30755662918091
L 90.86340665817261
L 91.21713638305664
L 91.17233753204346
L 91.37728810310364
L 90.77110290527344
L 90.64703583717346
L 91.17501378059387
L 91.59137010574341
L 91.52969717979431
L 91.10185503959656
L 90.52305817604065
L 90.97665548324585
L 90.73300957679749
L 91.0124659538269
L 91.4629340171814
L 90.9772276878357
L 91.66489243507385
L 90.93034267425537
L 91.28922820091248
L 91.65451526641846
L 91.61790609359741
L 90.96153974533081
L 91.19025468826294
L 91.43056273460388
L 90.83878993988037
L 90.47834277153015
```

(Fig. 14: Testing L output class)

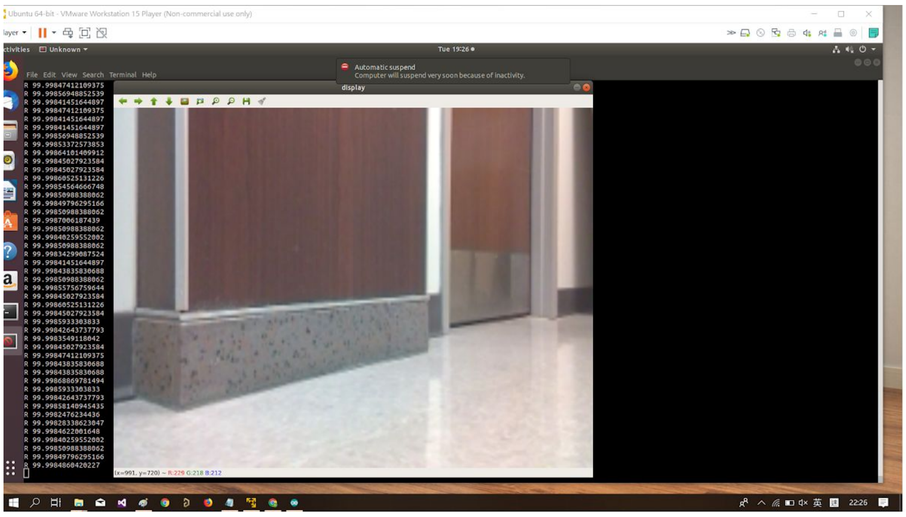


Automatic suspend
Computer will suspend very soon because of inactivity.

```
W 99.82299208641052
W 99.87435936927795
W 99.87000226974487
W 99.88487958908081
W 99.89431500434875
W 99.89311099052429
W 99.89387392997742
W 99.88773465156555
W 99.88909363746643
W 99.88659620285034
W 99.88777041435242
W 99.88734126091003
W 99.89019632339478
W 99.88880753517151
W 99.8903751373291
W 99.88879561424255
W 99.88852143287659
W 99.88847374916077
W 99.88777041435242
W 99.88773465156555
W 99.88722205162048
W 99.88647699356079
W 99.8861312866211
W 99.88890290260315
W 99.88922476768494
W 99.88721013069153
W 99.88489151000977
W 99.88631010055542
W 99.88957643508911
W 99.89100694656372
W 99.8874843120575
W 99.88816380500793
W 99.88903403282166
W 99.88728165626526
W 99.8875081539154
W 99.88621473312378
W 99.88911747932434
W 99.88738894462585
W 99.88817572593689
W 99.88654851913452
W 99.88471269607544
W 99.88881945610046
W 99.88678693771362
W 99.88793730735779
W 99.88559484481812
W 99.88712668418884
```

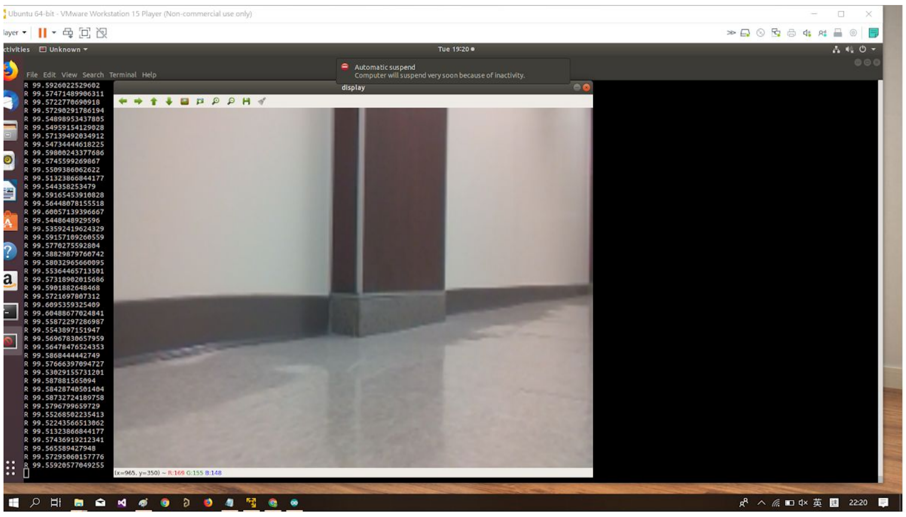
```
W 52.03786492347717
W 53.86136174201965
W 49.86797273159027
W 53.58855724334717
W 54.133015871047974
W 53.354525566101074
W 55.56954741477966
W 52.593791484832764
W 55.0248384475708
W 51.2134850025177
W 51.62619352340698
W 55.49545884132385
W 56.68225288391113
W 52.75040864944458
W 54.02169227600098
W 52.00210213661194
W 53.206849098205566
W 52.87649631500244
W 50.48344135284424
W 54.24054265022278
W 55.15453815460205
W 54.688018560409546
W 53.81396412849426
W 51.34068727493286
W 53.80733609199524
W 53.48765254020691
W 52.48095989227295
W 53.35935950279236
W 54.048675298690796
W 51.8659770488739
W 52.84487009048462
W 57.38754868507385
W 54.66269254684448
W 54.570215940475464
W 50.034159421920776
W 52.865177392959595
W 55.02161383628845
W 54.72787618637085
W 55.38443326950073
W 56.60218000411987
W 53.71091365814209
W 52.61463522911072
W 54.62964177131653
W 54.197269678115845
W 50.14386177062988
W 53.69511842727661
```

(Fig 15: Testing W output class)


Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Activities
Unknown
Tue 19:26
File Edit View Search Terminal Help
Automatic suspend
Computer will suspend very soon because of inactivity.
display
22:26

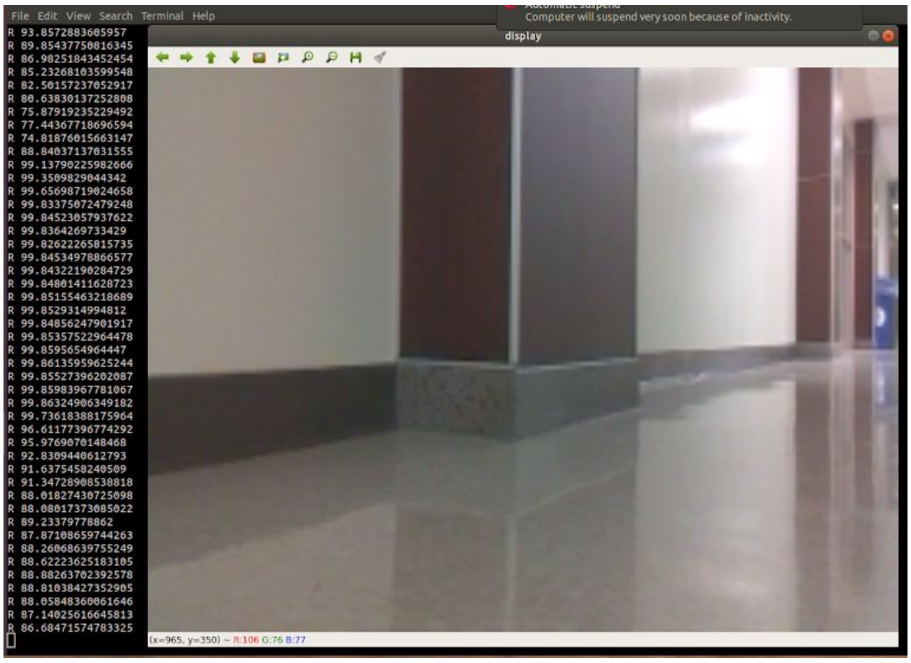
```
R 99.99847412109375
R 99.99856948852539
R 99.99841451644897
R 99.99847412109375
R 99.99841451644897
R 99.99841451644897
R 99.99856948852539
R 99.99853372573853
R 99.99864101409912
R 99.99845027923584
R 99.99845027923584
R 99.99860525131226
R 99.99854564666748
R 99.99850988388062
R 99.99849796295166
R 99.99850988388062
R 99.9987006187439
R 99.99850988388062
R 99.99840259552002
R 99.99850988388062
R 99.99834299087524
R 99.99841451644897
R 99.99843835830688
R 99.99850988388062
R 99.99855756759644
R 99.99845027923584
R 99.99860525131226
R 99.99845027923584
R 99.9985933303833
R 99.99842643737793
R 99.9983549118042
R 99.99845027923584
R 99.99847412109375
R 99.99843835830688
R 99.99843835830688
R 99.99868869781494
R 99.9985933303833
R 99.99842643737793
R 99.99858140945435
R 99.9982476234436
R 99.99828338623047
R 99.9984622001648
R 99.99840259552002
R 99.99850988388062
R 99.99849796295166
R 99.9984860420227
```

(x=991, y=720) ~ R:229 G:218 B:212


Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Activities
Unknown
Tue 19:20
File Edit View Search Terminal Help
Automatic suspend
Computer will suspend very soon because of inactivity.
display
22:20

```
R 99.592602252960 2
R 99.57471489906311
R 99.5722770690918
R 99.57290291786194
R 99.54898953437805
R 99.54959154129028
R 99.57139492034912
R 99.54734444618225
R 99.59800243377686
R 99.5745599269867
R 99.5509386062622
R 99.51323866844177
R 99.544358253479
R 99.59165453910828
R 99.56448078155518
R 99.60057139396667
R 99.5448648929596
R 99.53592419624329
R 99.59157109260559
R 99.5770275592804
R 99.58829879760742
R 99.58032965660095
R 99.55364465713501
R 99.57318902015686
R 99.5901882648468
R 99.5721697807312
R 99.6095359325409
R 99.60488677024841
R 99.55872297286987
R 99.5543897151947
R 99.56967830657959
R 99.56478476524353
R 99.5868444442749
R 99.57666397094727
R 99.53029155731201
R 99.587881565094
R 99.58428740501404
R 99.58732724189758
R 99.5796799659729
R 99.55268502235413
R 99.52243566513062
R 99.51323866844177
R 99.57436919212341
R 99.565589427948
R 99.57295060157776
R 99.55920577049255
```

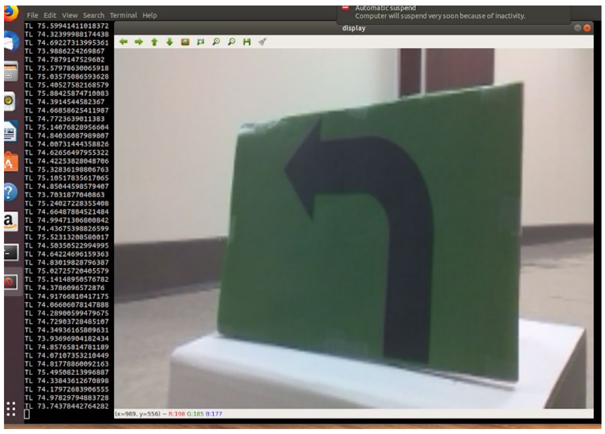
(x=965, y=350) ~ R:169 G:155 B:148


File Edit View Search Terminal Help
Computer will suspend very soon because of inactivity.
display

```
R 93.8572883605957
R 89.85437750816345
R 86.98251843452454
R 85.23268103599548
R 82.50157237052917
R 80.63830137252808
R 75.87919235229492
R 77.44367718696594
R 74.81876015663147
R 88.84037137031555
R 99.13790225982666
R 99.3509829044342
R 99.65698719024658
R 99.83375072479248
R 99.84523057937622
R 99.8364269733429
R 99.82622265815735
R 99.84534978866577
R 99.84322190284729
R 99.84801411628723
R 99.85155463218689
R 99.8529314994812
R 99.84856247901917
R 99.85357522964478
R 99.8595654964447
R 99.86135959625244
R 99.85527396202087
R 99.85983967781067
R 99.86324906349182
R 99.73618388175964
R 96.61177396774292
R 95.9769070148468
R 92.8309440612793
R 91.6375458240509
R 91.34728908538818
R 88.01827430725098
R 88.08017373085022
R 89.23379778862
R 87.87108659744263
R 88.26068639755249
R 88.62223625183105
R 88.88263702392578
R 88.81038427352905
R 88.05848360061646
R 87.14025616645813
R 86.68471574783325
```

(x=965, y=350) ~ R:106 G:76 B:77

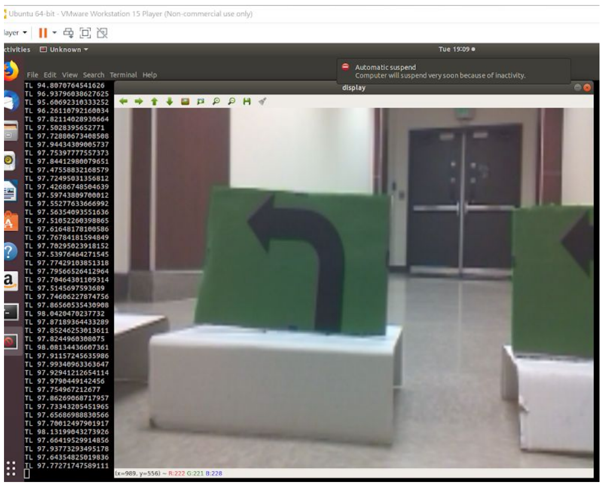
(Fig. 16: Testing R output class)



Automatic suspend
Computer will suspend very soon because of inactivity.

```
TL 75.59941411018372
TL 74.32399988174438
TL 74.69227313995361
TL 73.9886224269867
TL 74.7879147529602
TL 75.57978630065918
TL 75.03575086593628
TL 75.40527582168579
TL 75.88425874710083
TL 74.3914544582367
TL 74.66858625411987
TL 74.7723639011383
TL 75.14076828956604
TL 74.84036087989807
TL 74.00731444358826
TL 74.62656497955322
TL 74.42253828048706
TL 75.32836198806763
TL 75.10517835617065
TL 74.85044598579407
TL 73.7031877040863
TL 75.24027228355408
TL 74.66487884521484
TL 74.99471306800842
TL 74.43675398826599
TL 75.52313208580017
TL 74.50350522994995
TL 74.64224696159363
TL 74.83019828796387
TL 75.02725720405579
TL 75.14148950576782
TL 74.3786096572876
TL 74.91766810417175
TL 74.06606078147888
TL 74.28900599479675
TL 74.72903728485107
TL 74.34936165809631
TL 73.93696904182434
TL 74.85765814781189
TL 74.07107353210449
TL 74.81778860092163
TL 75.49508213996887
TL 74.33843612670898
TL 74.17972683906555
TL 74.97829794883728
TL 73.74378442764282
```

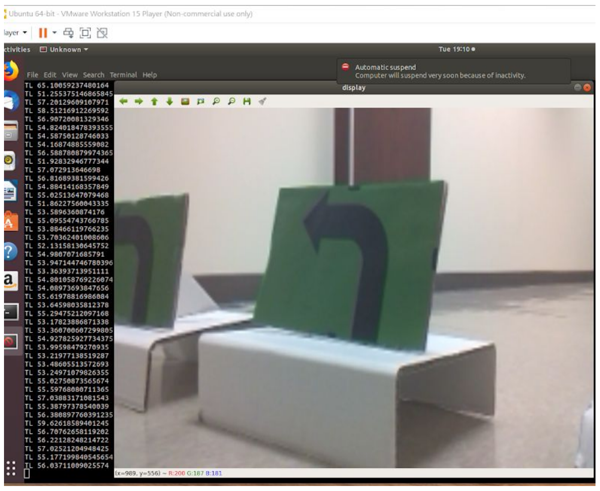
(x=989, y=556) ~ R:198 G:185 B:177


Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
layer
ctivities
Unknown
Tue 15:09
File Edit View Search Terminal Help
display

Automatic suspend
Computer will suspend very soon because of inactivity.

```
TL 94.8070764541626
TL 96.93796038627625
TL 95.60692310333252
TL 96.26110792160034
TL 97.82114028930664
TL 97.5028395652771
TL 97.72880673408508
TL 97.94434309005737
TL 97.75397777557373
TL 97.84412980079651
TL 97.47558832168579
TL 97.72495031356812
TL 97.42686748504639
TL 97.59743809700012
TL 97.55277633666992
TL 97.56354093551636
TL 97.51052260398865
TL 97.61648178100586
TL 97.76784181594849
TL 97.70295023918152
TL 97.53976464271545
TL 97.77429103851318
TL 97.79566526412964
TL 97.70464301109314
TL 97.5145697593689
TL 97.74606227874756
TL 97.86560535430908
TL 98.0420470237732
TL 97.87189364433289
TL 97.85246253013611
TL 97.8244960308075
TL 98.08134436607361
TL 97.91157245635986
TL 97.99340963363647
TL 97.92941212654114
TL 97.9790449142456
TL 97.754967212677
TL 97.86269068717957
TL 97.73343205451965
TL 97.65686988830566
TL 97.70012497901917
TL 98.13199043273926
TL 97.66419529914856
TL 97.93773293495178
TL 97.64354825019836
TL 97.77271747589111
```

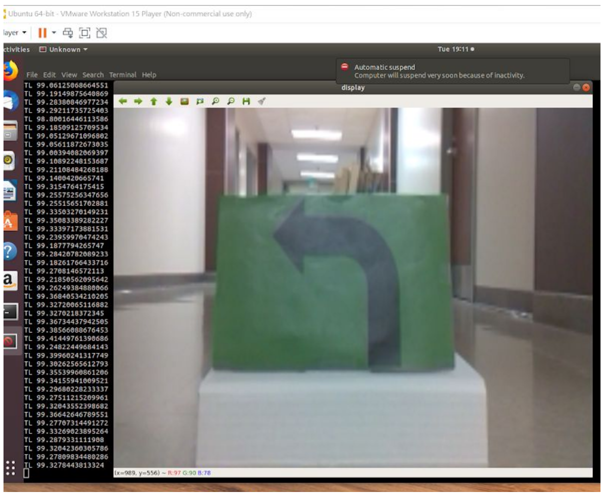
(x=989, y=556) ~ R:222 G:221 B:228


Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
layer
ctivities
Unknown
Tue 15:10
File Edit View Search Terminal Help
display

Automatic suspend
Computer will suspend very soon because of inactivity.

```
TL 65.10059237480164
TL 51.255375146865845
TL 57.20129609107971
TL 58.51216912269592
TL 56.90720081329346
TL 54.824018478393555
TL 54.58750128746033
TL 54.16874885559082
TL 56.588780879974365
TL 51.92832946777344
TL 57.072913646698
TL 56.81689381599426
TL 54.88414168357849
TL 55.02513647079468
TL 51.86227560043335
TL 53.5896360874176
TL 55.09554743766785
TL 53.88466119766235
TL 53.70362401008606
TL 52.13158130645752
TL 54.9807071685791
TL 53.947144746780396
TL 53.36393713951111
TL 54.801058769226074
TL 54.08973693847656
TL 55.61978816986084
TL 53.64598035812378
TL 55.29475212097168
TL 53.17823886871338
TL 53.360700607299805
TL 54.927825927734375
TL 53.99598479270935
TL 53.21977138519287
TL 53.48605513572693
TL 53.24971079826355
TL 55.02750873565674
TL 55.59768080711365
TL 57.03883171081543
TL 55.38797378540039
TL 56.380897760391235
TL 59.62618589401245
TL 56.70762658119202
TL 56.22128248214722
TL 57.02521204948425
TL 55.177199840545654
TL 56.03711009025574
```

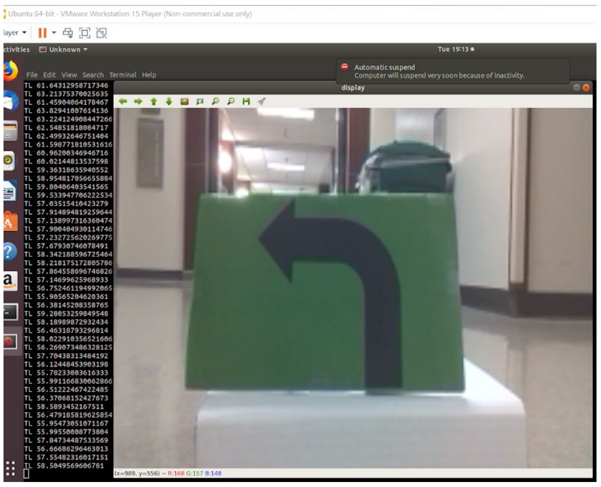
(x=989, y=556) ~ R:200 G:187 B:181


Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Activities
Unknown
Tue 19:11
File Edit View Search Terminal Help
display

Automatic suspend
Computer will suspend very soon because of inactivity.

```
TL 99.06125068664551
TL 99.19149875640869
TL 99.28380846977234
TL 99.29211735725403
TL 98.80016446113586
TL 99.18509125709534
TL 99.05129671096802
TL 99.05611872673035
TL 99.00394082069397
TL 99.10892248153687
TL 99.21108484268188
TL 99.1400420665741
TL 99.3154764175415
TL 99.25575256347656
TL 99.25515651702881
TL 99.33503270149231
TL 99.35083389282227
TL 99.33397173881531
TL 99.23959970474243
TL 99.1877794265747
TL 99.28420782089233
TL 99.18261766433716
TL 99.2708146572113
TL 99.21850562095642
TL 99.26249384880066
TL 99.36840534210205
TL 99.32720065116882
TL 99.3270218372345
TL 99.36734437942505
TL 99.38566088676453
TL 99.41449761390686
TL 99.24822449684143
TL 99.39960241317749
TL 99.30262565612793
TL 99.35539960861206
TL 99.34155941009521
TL 99.29680228233337
TL 99.27511215209961
TL 99.32043552398682
TL 99.36642646789551
TL 99.27707314491272
TL 99.33269023895264
TL 99.2879331111908
TL 99.32042360305786
TL 99.27809834480286
TL 99.3278443813324
```

(x=989, y=556) ~ R:97 G:90 B:78

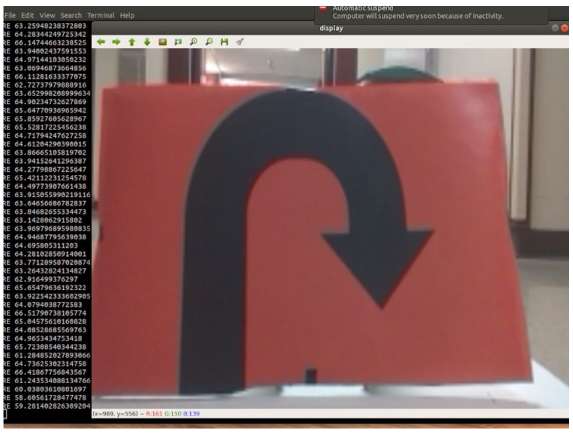

Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Activities
Unknown
Tue 19:13
File Edit View Search Terminal Help
display

Automatic suspend
Computer will suspend very soon because of inactivity.

```
TL 61.64312958717346
TL 63.21375370025635
TL 61.45904064178467
TL 63.82941007614136
TL 63.224124908447266
TL 62.54851818084717
TL 62.49932646751404
TL 61.598771810531616
TL 60.96200346946716
TL 60.02144813537598
TL 59.36318635940552
TL 58.954817056655884
TL 59.80406403541565
TL 59.533947706222534
TL 57.03515410423279
TL 57.914894819259644
TL 57.138997316360474
TL 57.900404930114746
TL 57.232725620269775
TL 57.67930746078491
TL 58.342188596725464
TL 58.218175172805786
TL 57.864558696746826
TL 57.14699625968933
TL 56.752461194992065
TL 55.90565204620361
TL 56.38145208358765
TL 59.28053259849548
TL 58.18989872932434
TL 56.46318793296814
TL 58.022910356521606
TL 56.269073486328125
TL 57.70438313484192
TL 56.12448453903198
TL 55.78233003616333
TL 55.991166830062866
TL 56.51222467422485
TL 56.37068152427673
TL 58.5893452167511
TL 56.479185819625854
TL 55.95473051071167
TL 55.99550008773804
TL 57.84734487533569
TL 56.66686296463013
TL 57.55482316017151
TL 58.5049569606781
```

(x=989, y=556) ~ R:168 G:157 B:148

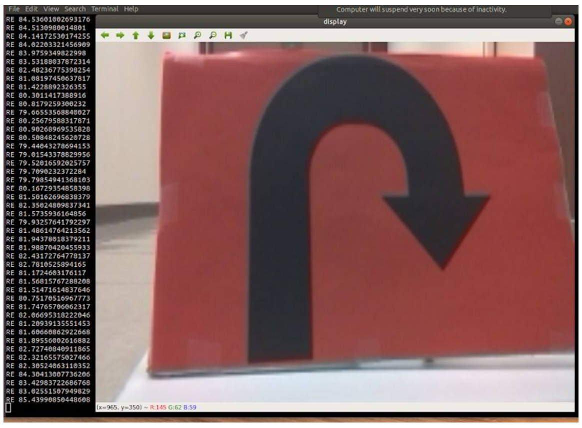
(Fig 17: Testing TL output class)



Automatic suspend
Computer will suspend very soon because of inactivity.

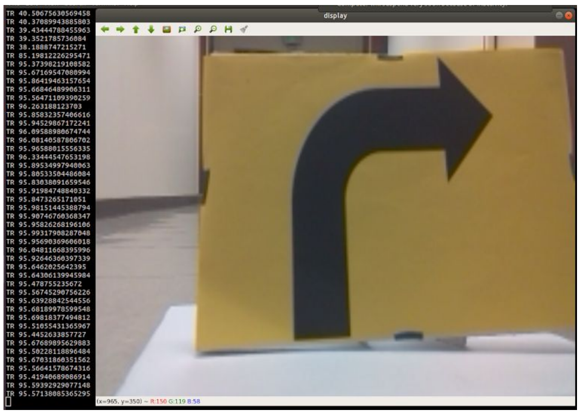
```
RE 63.25948238372803
RE 64.28344249725342
RE 66.14744663238525
RE 63.94002437591553
RE 64.97144103050232
RE 63.06946873664856
RE 66.11281633377075
RE 62.72737979888916
RE 63.652998208999634
RE 64.90234732627869
RE 65.64770936965942
RE 65.85927605628967
RE 65.52817225456238
RE 64.71794247627258
RE 64.61204290390015
RE 63.86665105819702
RE 63.94152641296387
RE 64.27798867225647
RE 65.42112231254578
RE 64.49773907661438
RE 63.915055990219116
RE 63.64656686782837
RE 63.84682655334473
RE 63.1428062915802
RE 63.969796895980835
RE 64.94687795639038
RE 64.695805311203
RE 64.28102850914001
RE 63.771289587020874
RE 63.26432824134827
RE 62.916499376297
RE 65.65479636192322
RE 63.922542333602905
RE 64.0794038772583
RE 66.51790738105774
RE 65.04575610160828
RE 64.08528685569763
RE 64.9653434753418
RE 65.72308540344238
RE 61.284852027893066
RE 64.73625302314758
RE 66.41867756843567
RE 61.243534088134766
RE 60.03803610801697
RE 58.60561728477478
RE 59.281402826309204
```

(x=989, y=556) ~ R:161 G:150 B:139


File Edit View Search Terminal Help
Computer will suspend very soon because of inactivity.
display
(x=965, y=350) ~ R:145 G:62 B:59

```
RE 84.53601002693176
RE 84.5130980014801
RE 84.14172530174255
RE 84.02203321456909
RE 83.9759349822998
RE 83.53188037872314
RE 82.48236775398254
RE 81.08197450637817
RE 81.4228892326355
RE 80.3011417388916
RE 80.8179259300232
RE 79.66553568840027
RE 80.25679588317871
RE 80.90268969535828
RE 80.50848245620728
RE 79.44043278694153
RE 79.01543378829956
RE 79.52016592025757
RE 79.7090232372284
RE 79.79854941368103
RE 80.16729354858398
RE 81.50162696838379
RE 82.35024809837341
RE 81.5735936164856
RE 79.93257641792297
RE 81.48614764213562
RE 81.94378018379211
RE 81.98870420455933
RE 82.43172764778137
RE 82.7810525894165
RE 81.1724603176117
RE 81.56815767288208
RE 81.51471614837646
RE 80.75170516967773
RE 81.74765706062317
RE 82.06695318222046
RE 81.20939135551453
RE 81.60660862922668
RE 81.89556002616882
RE 82.72740840911865
RE 82.32165575027466
RE 82.30524063110352
RE 84.30413007736206
RE 83.42983722686768
RE 83.02551507949829
RE 85.43990850448608
```

(Fig 18: Testing RE output class)



```
TR 40.50675630569458
TR 40.37089943885803
TR 39.43444788455963
TR 39.3521785736084
TR 38.1888747215271
TR 85.19812226295471
TR 95.37398219108582
TR 95.67169547080994
TR 95.86419463157654
TR 95.66846489906311
TR 95.56471109390259
TR 96.263188123703
TR 95.85832357406616
TR 95.94529867172241
TR 96.09588980674744
TR 96.08140587806702
TR 95.96588015556335
TR 96.33444547653198
TR 95.89534997940063
TR 95.80533504486084
TR 95.83038091659546
TR 95.91984748840332
TR 95.8473265171051
TR 95.98151445388794
TR 95.90746760368347
TR 95.95826268196106
TR 95.99317908287048
TR 95.95690369606018
TR 96.04811668395996
TR 95.92646360397339
TR 95.6462025642395
TR 95.64306139945984
TR 95.478755235672
TR 95.56745290756226
TR 95.63928842544556
TR 95.68189978599548
TR 95.69818377494812
TR 95.51055431365967
TR 95.4452633857727
TR 95.67689895629883
TR 95.50228118896484
TR 95.67031860351562
TR 95.56641578674316
TR 95.41940689086914
TR 95.59392929077148
TR 95.57138085365295
```

(Fig. 19: Testing TR output class)

8. Analysis

Despite frequent data collection and training, the accuracy and usability of the network throughout much of the experiment were not very high. There are a number of factors which play into this. One of these is the fact that the hallway in which the car was trained is a bit lacking in distinct details to help guide it and recognize its location. The walls, floors, and doors were essentially all identical and the distinctive features such as recycling bins and snack area are not at spots that have any distinctive function for the car to perform. In addition, collecting and sorting the data posed some problems. One of these was that Google Drive, while the most effective means for sorting large numbers of images, can be slow and inefficient. Thus, the first few data collections, though a bit smaller, still took a long time to sort. Another issue with Google Drive is that sometimes when large downloads are made, some of the files are lost to condense the download, which led to some images being lost before the group realized this detail. In addition, some of the trainings early on and when new output classes were introduced did not have enough images in all the output classes, which led to the determination partway through that upwards of 1000 images for each class and a balanced number in each (such that no output class has upwards of twice as many as the others) were essential for consistent accuracy.

Furthermore, the Jetson Nano's limited amount of memory created some limitations on the run speed of the network. It only had about 2 GB of memory free to drive the car around. Thus, when given the weighty task of receiving images from the camera, putting them through the neural network, generating an output class, and giving the Arduino a command based on that output class, it sometimes ran slowly and lagged.

9. Future Works

In the future, we would like to change a couple of parameters to improve the performance of the car. The hallway the car was trained in had columns, diverse lighting, and objects such as chairs and recycling bins. All of these seemingly random patterns confuse the neural network and result in low accuracy and poor results. If we could utilize a hallway free of any distractions or one with explicit path boundaries, the results would be much better. Another method of improving the neural network's performance would be to label specific features contained in each image with object location networks. The process of assigning one label to an entire image is too broad to allow for the network to reliably pick the correct pattern. Therefore, the use of labeling sections of images, such as the corner between the floor and the wall, would cause an increase in driving accuracy. The modification which would likely show the most improvement is the addition of sensors and positional data. Any relevant data will improve the accuracy of the neural network. We could improve the Pi Camera with a wide angle lens or add an ultrasonic proximity sensor, but cost and practicality must always be considered.

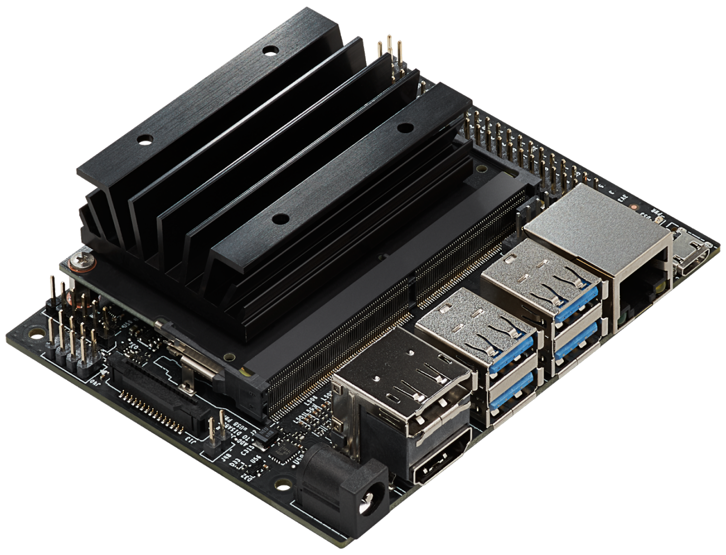
10. Conclusion

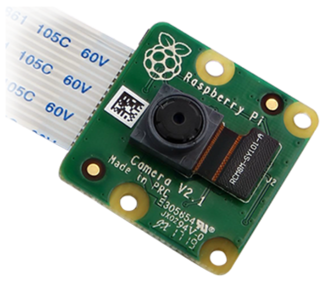
The conclusion of this experiment is that the Jetson Nano and MagmaDNN are suitable for controlling a self-driving miniature car, with some limitations. MagmaDNN would most likely require further development to allow models to be saved and the Jetson Nano would require some extremely careful memory allocation and research into the possibility of optimizing its performance.

Acknowledgements

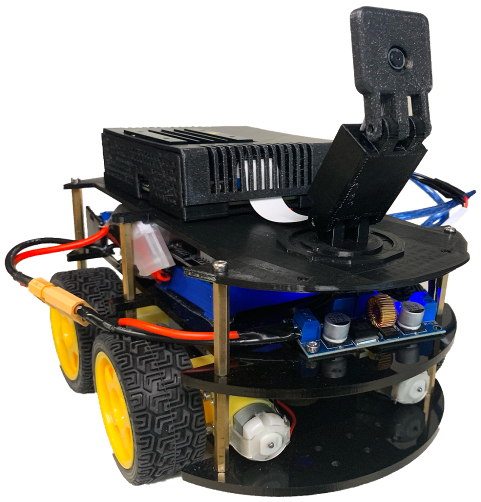
This project was sponsored by the National Science Foundation through the Research Experiences for Undergraduates (REU) award. This project was mentored by Dr. Alan Ayala (UTK) and Dr. Kwai Wong (UTK). This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE).

Appendix

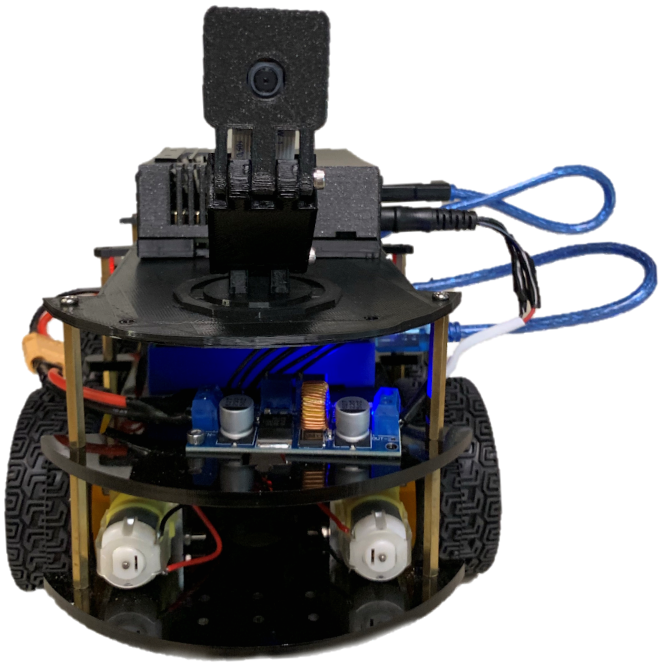

(Fig. 20: Jetson Nano Developer Kit)


(Fig. 21: Raspberry Pi Camera Module V2)

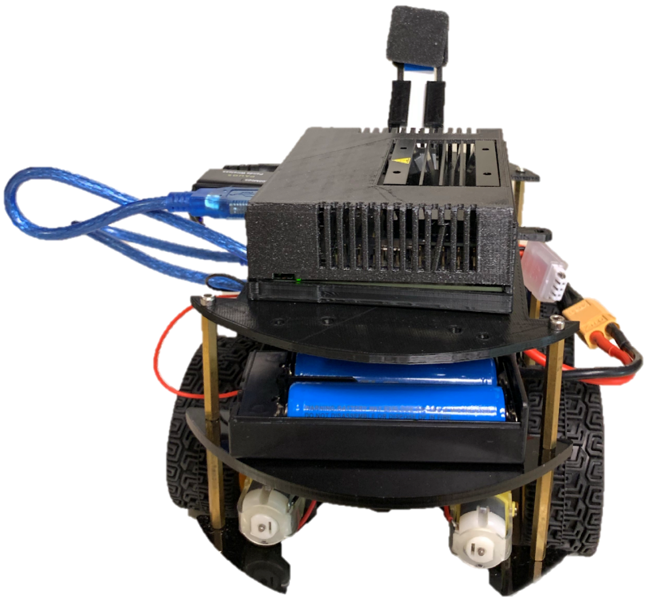
(Fig. 22: Wi-Fi Dongle)



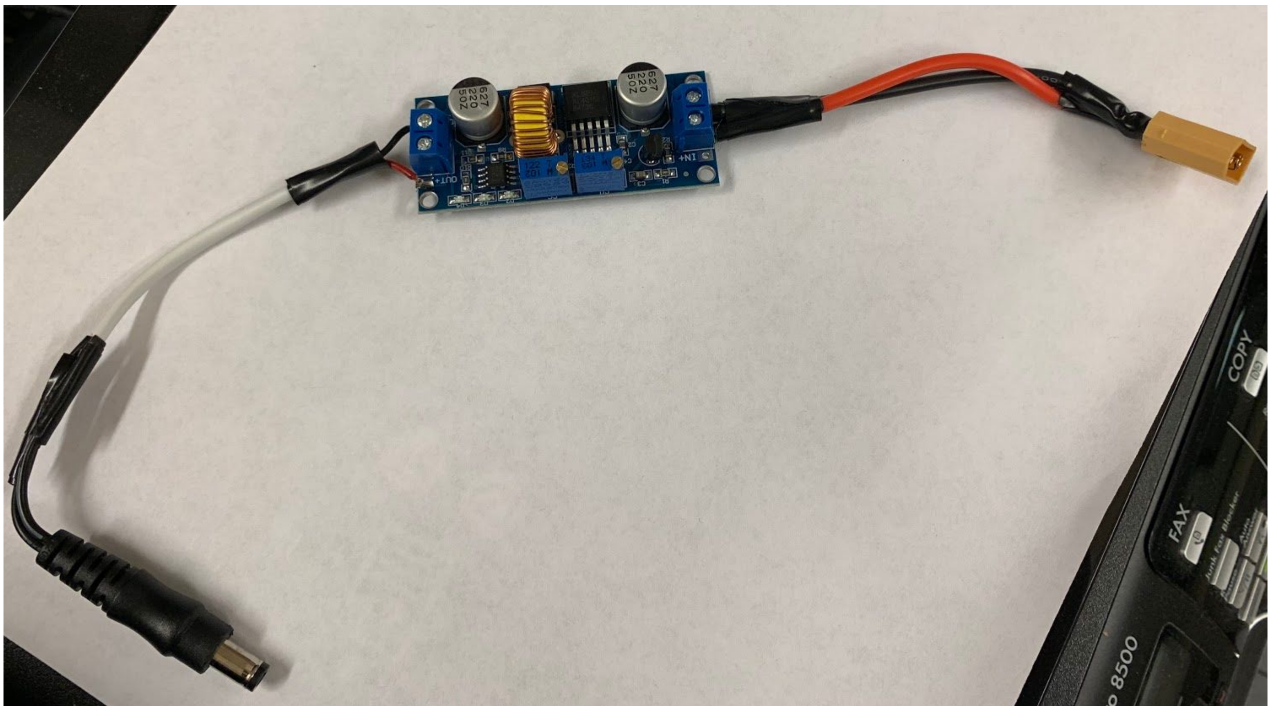
(Fig. 23: Overview of the Jetson Nano Car)



(Fig. 24: Front View of the Car)



(Fig. 25: Back view of the Car)


(Fig. 26: Voltage Regulator Module)

(Fig. 27: Battery for the Jetson Nano)

Python3 Code for Self-Driving

```
# ====================================================
from imageai.Prediction.Custom import ModelTraining
from imageai.Prediction import ImagePrediction


import os
import numpy as np
import cv2
import serial
import time

# ==========
# Prediction
# ==========
from imageai.Prediction.Custom import CustomImagePrediction


prediction = CustomImagePrediction()

# Create a prediction algorithm object
prediction.setModelTypeAsResNet()
#prediction.setModelTypeAsSqueezeNet()
#prediction.setModelTypeAsInceptionV3()
#prediction.setModelTypeAsDenseNet()

# Load prediction model (don't need to retrain)
execution_path = os.getcwd()
prediction.setModelPath(os.path.join(execution_path, "7-30-test/model.h5") )
prediction.setJsonPath( os.path.join(execution_path, "7-30-test/model_class.json") )


#=============
# define here
#=============
# Camera

def gstreamer_pipeline (capture_width=256, capture_height=144, display_width=256, display_height=144, framerate=2,
flip_method=2) :
        return ('nvarguscamerasrc ! '
        'video/x-raw(memory:NVMM), '
        'width=(int)%d, height=(int)%d, '
        'format=(string)NV12, framerate=(fraction)%d/1 ! '
        'nvvidconv flip-method=%d ! '
        'video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! '
        'videoconvert ! '
        'video/x-raw, format=(string)BGR ! appsink'  %
(capture_width,capture_height,framerate,flip_method,display_width,display_height))

cap = cv2.VideoCapture(gstreamer_pipeline(flip_method=2), cv2.CAP_GSTREAMER)

exPrediction =  ''
cuPrediction = ''
counter = 0
prob = 0
turn = 0

# Arduino
with serial.Serial('/dev/ttyACM0', 9600, timeout=10) as ser:

# ===================
# begin loop for saving images, running prediction, and outputting command to arduino
# ===================
        while True:

        ret, frame = cap.read()
        if(turn == turn):
        if ret==True:
        cv2.imwrite('image.jpg', frame)
        # Specifies display size
        #display = cv2.resize(frame,(426,240))
        #cv2.imshow('display',display)
        if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
        break

        #prediction.loadModel(num_objects=4, prediction_speed="fastest")
        prediction.loadModel(num_objects=6, prediction_speed="fastest")
```

```
predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "image.jpg"), result_count=1)
for eachPrediction, eachProbability in zip(predictions, probabilities):
print(eachPrediction, eachProbability)

exPrediction = cuPrediction
#print("Previous-Prediction = ", exPrediction)
cuPrediction = eachPrediction
#print("Current-Prediction = ", cuPrediction)

if exPrediction == cuPrediction:
counter += 1
#print("Counter = ", counter)
exPrediction = cuPrediction
else:
counter = 0

prob = float(eachProbability)
prob = round(prob)

if (prob > 30):
if eachPrediction == 'W':
        ser.write(bytes('W\n','utf-8'))
        turn = 0
        print('forward')


if  eachPrediction == 'L':
        ser.write(bytes('A\n','utf-8'))
        turn = 0
        print('left')


if  eachPrediction == 'R':
        ser.write(bytes('D\n','utf-8'))
        turn = 0
        print('right')


if  eachPrediction == 'TL':
        ser.write(bytes('R\n', 'utf-8'))
        print('turn left')
                #ser.write(bytes('W\n','utf-8'))
        #time.sleep(0.5)

if  eachPrediction == 'TR':
        turn = 0
        ser.write(bytes('Y\n', 'utf-8'))
        #ser.write(bytes('Y\n','utf-8'))
        print('turn right')
        #time.sleep(0.5)
        turn = 0
        #ser.write(bytes('W\n','utf-8'))
        #time.sleep(0.5)

if  eachPrediction == 'RE':
        if turn == 0:
        turn = 1
        ser.write(bytes('T\n','utf-8'))
        print('Return')
        #time.sleep(1)
        #ser.write(bytes('W\n','utf-8'))

        else:
```

```
                        ser.write(bytes('W\n','utf-8'))



ser.write(bytes('Q\n','utf-8'))
# Release everything if job is finished
cap.release()
cv2.destroyAllWindows
```

Arduino Code for motor control

```
/*define logic control output pin*/
//For Car 2.0
int in1=6;
int in2=7;
int in3=8;
int in4=9;
/*define channel enable output pins*/
int ENA=5;
int ENB=11;


/*Slow Speed of Car*/
int Sspd_L = 50;
int Sspd_R = 50;

/*Normal Speed of Car (0-255) */
int Nspd_L = 65;
int Nspd_R = 65;

/*Fast Speed of Car*/
int Fspd_L = 135;
int Fspd_R = 135;

/*Fast Fast Speed of Car*/
int FFspd_L = 255;
int FFspd_R = 255;
void setup() {

 /*Open the serial port and set the baud rate to 9600 */
 Serial.begin(9600);

/*Set the defined pins to the output*/
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);

  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);

/* wait for serial port to connect. */
  while (!Serial) {
    ;
}
}
/*define forward function*/
void _mForward_N()
{
  analogWrite(ENA, Nspd_L);
  analogWrite(ENB, Nspd_R);
  digitalWrite(in1,HIGH);//digital output
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
  /*Serial.println("Forward");*/
}

/*define back function*/
void _mBack()
{
 analogWrite(ENA, Nspd_R);
  analogWrite(ENB, Nspd_L);
  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
  /*Serial.println("Back");*/
}

/*define left function*/
void _mleft()
{
  analogWrite(ENA, Fspd_R);
  analogWrite(ENB, Fspd_L);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
  /*Serial.println("Left"); */
}

/*define right function*/
void _mright()
{
  analogWrite(ENA, Fspd_R);
  analogWrite(ENB, Fspd_L);
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
  /*Serial.println("Right"); */
}

void _mTR()
{
  analogWrite(ENA, FFspd_R);
  analogWrite(ENB, FFspd_L);
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
}

void _mTL()
{
  analogWrite(ENA, FFspd_R);
  analogWrite(ENB, FFspd_L);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
}

void _mReturn()
{
  analogWrite(ENA, FFspd_R);
  analogWrite(ENB, FFspd_L);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
  }

void _mStop()
{
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);

  /*Serial.println("Left"); */
}


void loop() {
  char buffer[16];

  /*if we get a command */
  if (Serial.available() > 0) {
    int size = Serial.readBytesUntil('\n', buffer, 12);
    if (buffer[0] == 'W') {
      _mForward_N();
    }

    if (buffer[0] == 'A') {
      _mleft();
    }

    if (buffer[0] == 'S') {
      _mBack();
    }

    if (buffer[0] == 'D') {
      _mright();
    }

    if (buffer[0] == 'Q'){
      _mStop();
    }

    if (buffer[0] == 'T'){
     _mReturn();
    }

    if (buffer[0] == 'R'){
     _mTL();

  }

    if (buffer[0] == 'Y'){
     _mTR();


  }
}
```

```
}
```

References

[1] Inzaugarat, Euge (2018). Understanding Neural Networks: What, How and Why? Retrieved from https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31
[2] Olafenwa, Moses (2018). Train Image Recognition AI with 5 lines of code. Retrieved from https://towardsdatascience.com/train-image-recognition-ai-with-5-lines-of-code-8ed0bdd8d9ba