Autonomous Vehicle Research Project

Members:

Patrick Lau

Julian Halloy

Brendan Flood

Mentor:

Dr. Alan Ayala

Objectives 

- Implementation of MagmaDNN for image recognition
- To use trained neural networks to control the motion of a self-driving car consisting of:
  - A Jetson Nano computer with built in GPU
  - Elegoo Car kit with Arduino UNO board
  - A Raspberry Pi camera to collect image data input

Research Plan 

Steps:

1. Test ImageAI networks to begin to establish self-driving abilities
2. Train and test MagmaDNN networks to meet benchmarks and drive the car
    a. Incorporate input image reading
3. Improve networks to maximize accuracy while maintaining high enough run speed
4. Train with additional images to meet different benchmarks

Math:

- Neural Networks: Matrix multiplication, activation functions, back propagation of weights from loss values

Research Plan (cont.)

Algorithm:

- For the robot
  - Take input from the Raspberry Pi camera which is delivered to the Jetson Nano
  - Use previously trained neural network model to identify images
  - Based on the identity of photo input, Jetson nano sends a command to the Arduino UNO to control its movement in response
  - Process repeats with additional images that the camera takes, until the program is killed or a stated goal is met
- Within the neural networks
  - Layers: Pairs of convolutional layers followed by pooling layers, RELU activation function, finishes with a flatten layer and a dense layer, loss functions and weight decay TBD
  - Input: a set of hand-collected images which is saved in a directory containing a test and train directory, each of which contains directories of images whose names correspond to the labels of the images in them
  - Filesystem recursive_directory_iterator is used to find images within these directories and fill a labels vector with their names, opencv is used to convert them to pixel matrices, and these values are then read into training tensors

Research Plan (cont.)

Benchmarks:

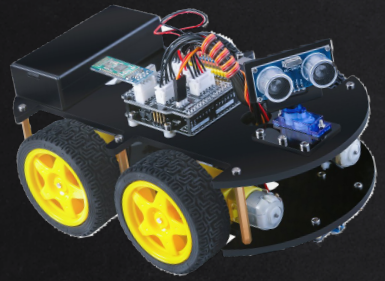
1. Basic self-driving capability - whether or not the car can navigate the hall by avoiding hitting walls and turning down other hallways when told to

2. Sign recognition - whether or not the car can recognize signs in the hallway (e.g. posters with numbers on them) and respond to them as commanded

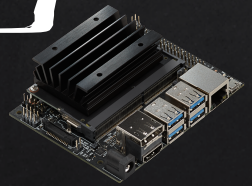
3. Following - whether or not a car can recognize a second car driving in the hallway and follow it

Working process

Construct the car

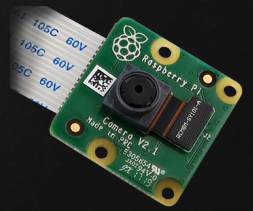


Setup the Jetson Nano



Jetson Nano Car

Install the Camera



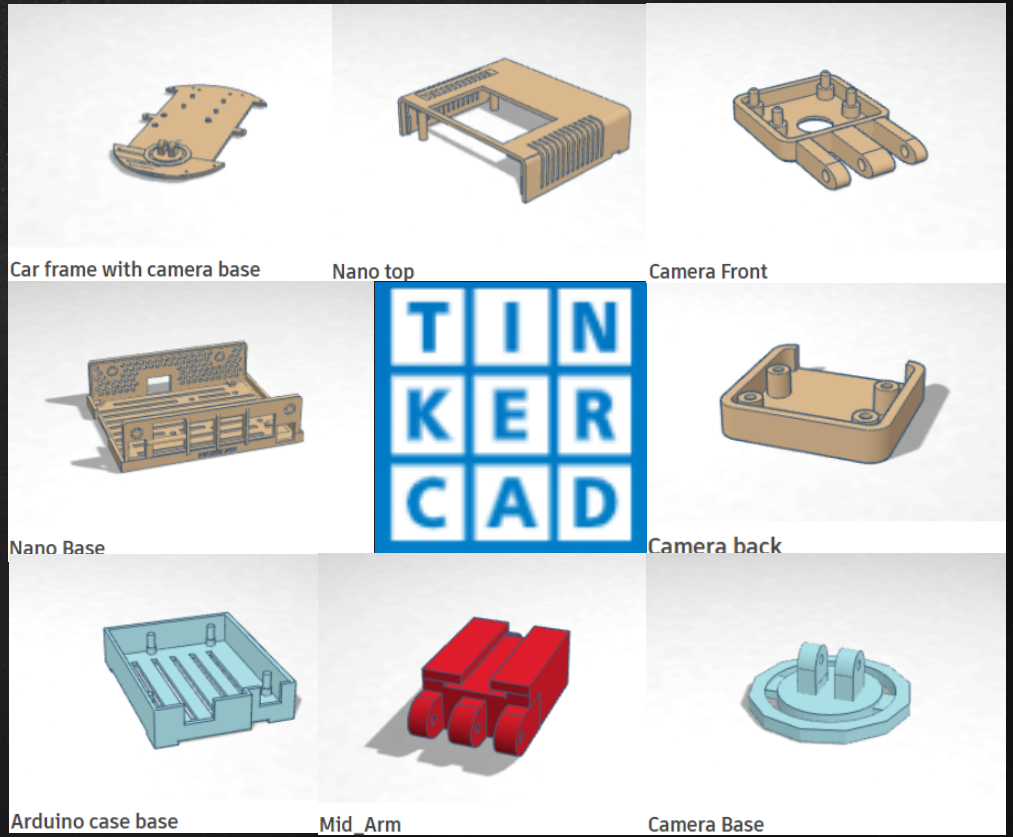
Collect Data

Labeling

Model Training

3D printing

- Combine every component together
- Why TinkerCAD?
  - Free
  - Online, no installation
  - Low hardware requirement
  - Easy to popularize



Nano & Uno Communication

- .py & .io
- PySerial
  - Envelop the access for the serial port
  - Communication between the Arduino board and JetsonNano
  - read()
  - write()

```python
import serial

with serial.Serial('/dev/ttyACM0', 9600, timeout=10) as ser:
    while True:
        move = input('')[0]
        if move in 'wW':
            ser.write(bytes('W\n','utf-8'))
            print('forward')
        if move in 'sS':
            ser.write(bytes('S\n','utf-8'))
            print('backward')
        if move in 'aA':
            ser.write(bytes('A\n','utf-8'))
            print('left')
        if move in 'dD':
            ser.write(bytes('D\n','utf-8'))
            print('right')
        if move in 'qQ':
            ser.write(bytes('Q\n','utf-8'))
            print('stop')
        if move in 'xX':
            ser.write(bytes('X\n','utf-8'))
            print('slow down')
        if move in 'zZ':
            ser.write(bytes('Z\n','utf-8'))
            print('speed up')
```

```python
        #prediction.loadModel(num_objects=4, predictio
        prediction.loadModel(num_objects=4)
        predictions, probabilities = prediction.predic
        for eachPrediction, eachProbability in zip(pre
            print(eachPrediction, eachProbability)


        if eachPrediction == 'W':
            ser.write(bytes('W\n','utf-8'))
            print('forward')
        if eachPrediction == 'B':
            ser.write(bytes('S\n','utf-8'))
            print('backward')
        if eachPrediction == 'L':
            ser.write(bytes('A\n','utf-8'))
            print('left')
        if eachPrediction == 'R':
            ser.write(bytes('D\n','utf-8'))
            print('right')

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

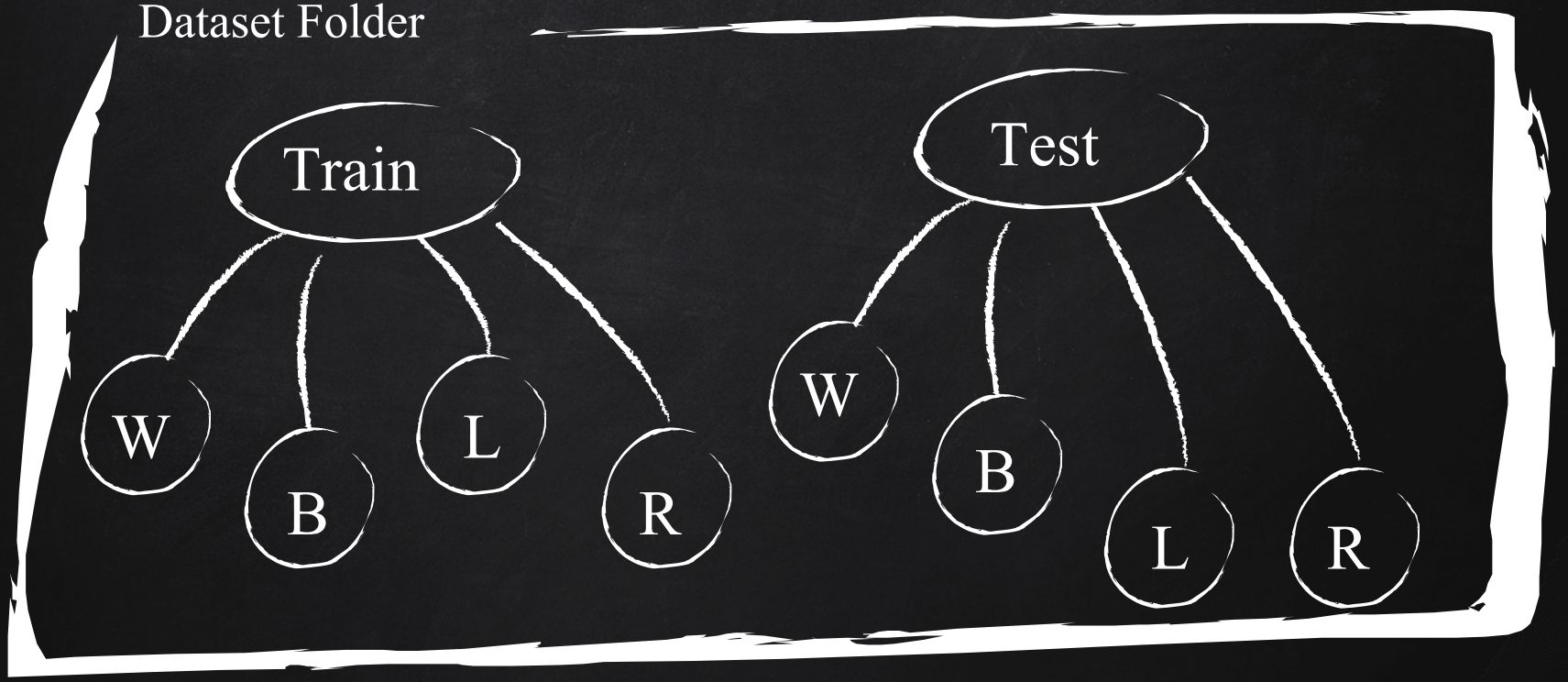
```arduino
void loop() {
  char buffer[16];

  /*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();
    }
```

Data Collection

```python
import numpy as np
import cv2
import os

def gstreamer_pipeline (capture_width=1920, capture_height=1080, display_width=1920, display_heig
ht=1080, framerate=15, 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_m
ethod,display_width,display_height))


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


num = 0
while os.path.exists('images{}'.format(num)):
    num += 1


# Create file path for images
dirName = 'images{}'.format(num)
os.mkdir(dirName)
print("Directory " , dirName ,  " Created ")

img_num = 0

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        cv2.imwrite('%s/image%d.jpg' % (dirName,img_num),frame)
        print('frame captured%d' % (img_num))
        img_num += 1
        # Specifies display size
        #display = cv2.resize(frame,(640,480))
        #cv2.imshow('display',display)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            print('Pressed Q')
            break
    else:
        break
```

```
File  Edit  View  Search  Terminal
captured image465.jpg
captured image466.jpg
captured image467.jpg
captured image468.jpg
captured image469.jpg
captured image470.jpg
captured image471.jpg
captured image472.jpg
captured image473.jpg
captured image474.jpg
captured image475.jpg
captured image476.jpg
captured image477.jpg
captured image478.jpg
captured image479.jpg
captured image480.jpg
captured image481.jpg
captured image482.jpg
captured image483.jpg
captured image484.jpg
captured image485.jpg
captured image486.jpg
captured image487.jpg
captured image488.jpg
captured image489.jpg
captured image490.jpg
captured image491.jpg
captured image492.jpg
captured image493.jpg
captured image494.jpg
captured image495.jpg
captured image496.jpg
captured image497.jpg
captured image498.jpg
captured image499.jpg
captured image500.jpg
captured image501.jpg
captured image502.jpg
captured image503.jpg
captured image504.jpg
```

Model Training - Labeling



Model Training 

- Train folder
  - To be used to train the model
  - At least >500 images per object, >1000 is great
- Test folder
  - To be used to test the model as it trains
  - 100~200 images per object
- JSON file
  - Stores data structures
- .h5 file
  - Contains multidimensional arrays

Autonomous Car

nano@nano-1: ~/Desktop/autonomous-Car

File Edit View Search Terminal Help

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

        ret, frame = cap.read()
        if ret==True:
            cv2.imwrite('image.jpg', frame)
            # Specifies display size
            #display = cv2.resize(frame,(640,480))
            #print('test3')
            #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=4)
        predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "image.jpg"), result_count=1)
        for eachPrediction, eachProbability in zip(predictions, probabilities):
            print(eachPrediction, eachProbability)


        if eachPrediction == 'W':
            ser.write(bytes('W\n','utf-8'))
            print('forward')
        if eachPrediction == 'B':
            ser.write(bytes('S\n','utf-8'))
            print('backward')
        if eachPrediction == 'L':
            ser.write(bytes('A\n','utf-8'))
            print('left')
        if eachPrediction == 'R':
            ser.write(bytes('D\n','utf-8'))
            print('right')

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

-- INSERT -- 68,22 Bot

Testing Result

```
[sudo] password for nano:
nano@nano-1:~/Desktop/autonomous-Car$ python3 predict-with-arduino.py
2019-06-28 11:46:24.986492: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:931] ARM64 does not suppor
2019-06-28 11:46:24.986662: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with prop
name: NVIDIA Tegra X1 major: 5 minor: 3 memoryClockRate(GHz): 0.9216
pciBusID: 0000:00:00.0
totalMemory: 3.87GiB freeMemory: 2.20GiB
2019-06-28 11:46:24.986736: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devic
2019-06-28 11:46:27.449839: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect Strea
2019-06-28 11:46:27.449903: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-06-28 11:46:27.449924: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2019-06-28 11:46:27.450059: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow devic
, name: NVIDIA Tegra X1, pci bus id: 0000:00:00.0, compute capability: 5.3)
2019-06-28 11:46:55.985090: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran
may mean that there could be performance gains if more memory were available.
2019-06-28 11:46:56.681877: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran
may mean that there could be performance gains if more memory were available.
2019-06-28 11:46:57.695281: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran
may mean that there could be performance gains if more memory were available.
2019-06-28 11:46:57.791171: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran
may mean that there could be performance gains if more memory were available.
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
```

```
2019-06-28
may mean t
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
W 100.0
forward
```

Reference

Nvidia Jetson Nano - https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write
MAGMA - https://icl.utk.edu/magma/software/index.html
MAGMADNN - https://magmadnn.bitbucket.io/docs/index.htmlhttps://magmadnn.bitbucket.io/docs/index.html
Arduino Connection - https://blog.rareschool.com/2019/05/five-steps-to-connect-jetson-nano-and.html
TinkerCAD - https://www.tinkercad.com
Pi Camera - https://github.com/JetsonHacksNano/CSI-Camera
Github, jkjung - https://github.com/jkjung-avt/jetson_nano
OpenCV - https://opencv.org/
TensorFlow - https://www.tensorflow.org/install/pip
Gstreamer - https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html?gi-language=c
OpenBLAS - https://www.openblas.net/
ImageAI - https://github.com/OlafenwaMoses/ImageAI
Elegoo - https://www.elegoo.com/download/
Arduino - https://www.arduino.cc/en/Main/Software
Pyserial - https://pythonhosted.org/pyserial/
Numpy - https://www.numpy.org/
Thingiverse - https://www.thingiverse.com/
Keras - https://keras.io/

thanks!

Any questions?