Autonomous Vehicle Research Project

Members:

Patrick Lau

Julian Halloy

Brendan Flood

Mentor:

Dr. Ala

Dr. Kw

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

Benchmarks met?

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

1. 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

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

Neural Networks

ImageAI:

- Downloadable library with pre-built networks
- Several options, from which we used ResNet

MagmaDNN

- Convolutional network of 17 hidden layers
- Sets of 2 convolutions with activation, pooling, dropout

ImageAI

ImageAI Progress and Models

Wall avoidance

Wall avoidance and single turn

Wall avoidance and turn signs

Wall avoidance and colored turn signs

Following car, UT logo

ImageAI

MagmaDNN

- Limited time to implement it
- Difficulty with compatibility on the nano (CPU version, CuDNN)
- Final Result: medium convolutional network trained on the nano with the MNIST dataset
- Further work:
  - Developing model saving and loading abilities
  - Writing and testing code to take jpeg as input to create a dataset

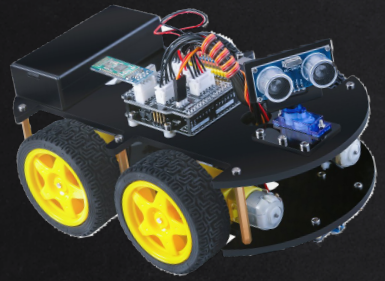
MagmaDNN (cont.)

```
    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,
6, a6, p3, d3, c7, a7, c8, a8, p4, d4, f, fc1, a9, fc2, a10, output_layer};

                                                                  216,1-8       89%
```

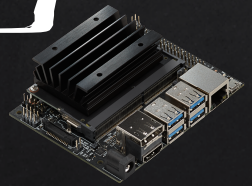
```
nano@nano-1:~$ ./first_network 7
Preparing to read 60000 images with size 28 x 28 ...
finished reading images.
Preparing to read 60000 labels with 10 classes ...
finished reading labels.
Image[7] is digit 3:
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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  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   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   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  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   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  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   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  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:~$
```

Working process

Construct the car

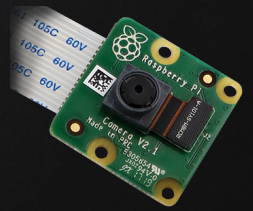


Setup the Jetson Nano



Jetson Nano Car

Install the Camera



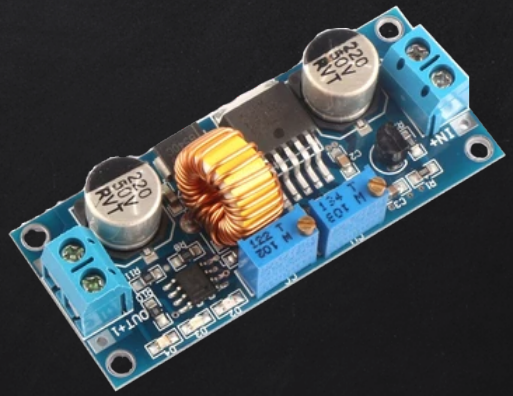
Collect Data

Labeling

Model Training

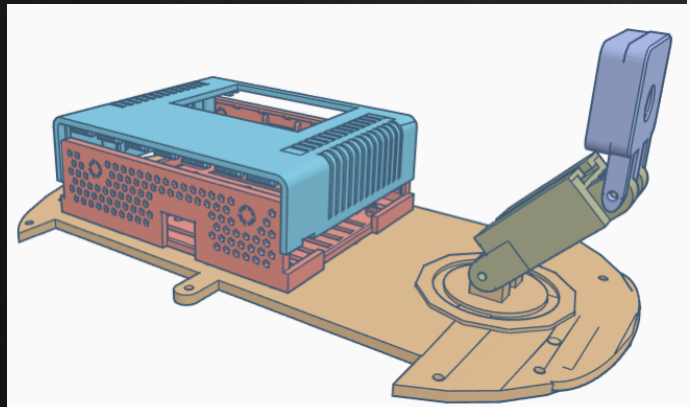
Power

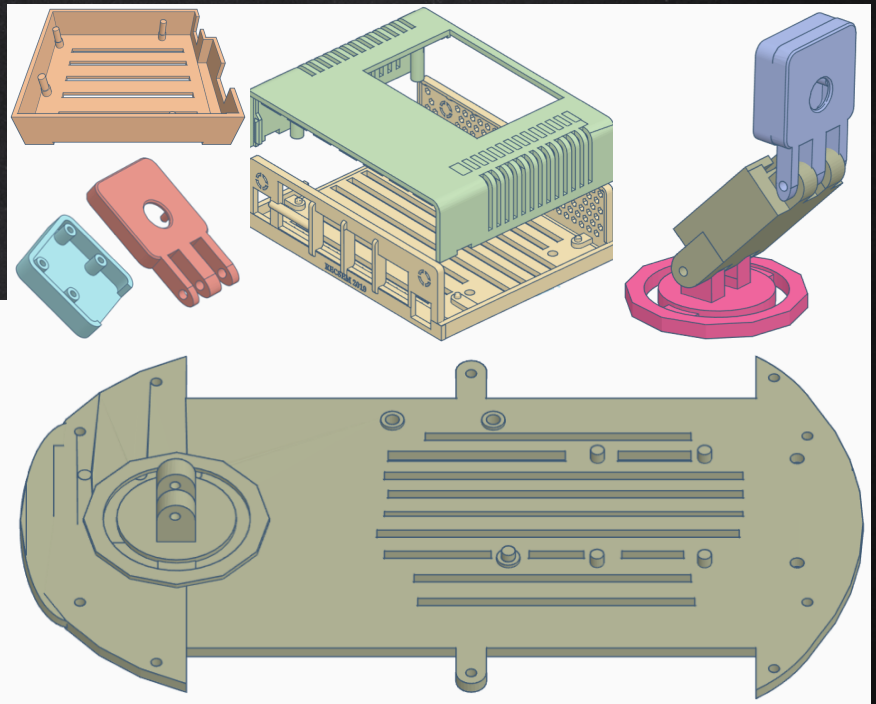
- Jetson Nano Power Modes: 5W & 10W
- Li-ion Battery: 5V 2A
- Lipo Battery: 11.1V 55A
  - Battery Management System
- VRM: 5V 4A



3D printing

- Combine every component together
- Protection





Nano & Uno Communication

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

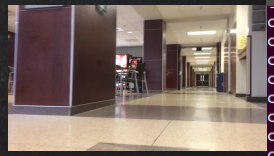
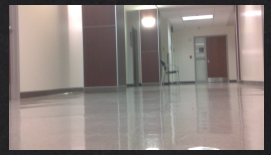
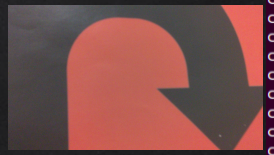
```python
import serial

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')
        if move in 'tT':
            ser.write(bytes('T\n','utf-8'))
            print('Return')
        if move in 'rR':
            ser.write(bytes('R\n','utf-8'))
            print('Left 90')
        if move in 'yY':
            ser.write(bytes('Y\n','utf-8'))
            print('Right 90')
```

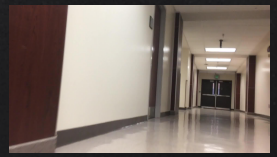
```arduino
void loop() {
  char buffer[16];

  /*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();
    }

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

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


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


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


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


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


    if  eachPrediction == 'RE':
        ser.write(bytes('T\n','utf-8'))
        print('Return')
```

Data Collection

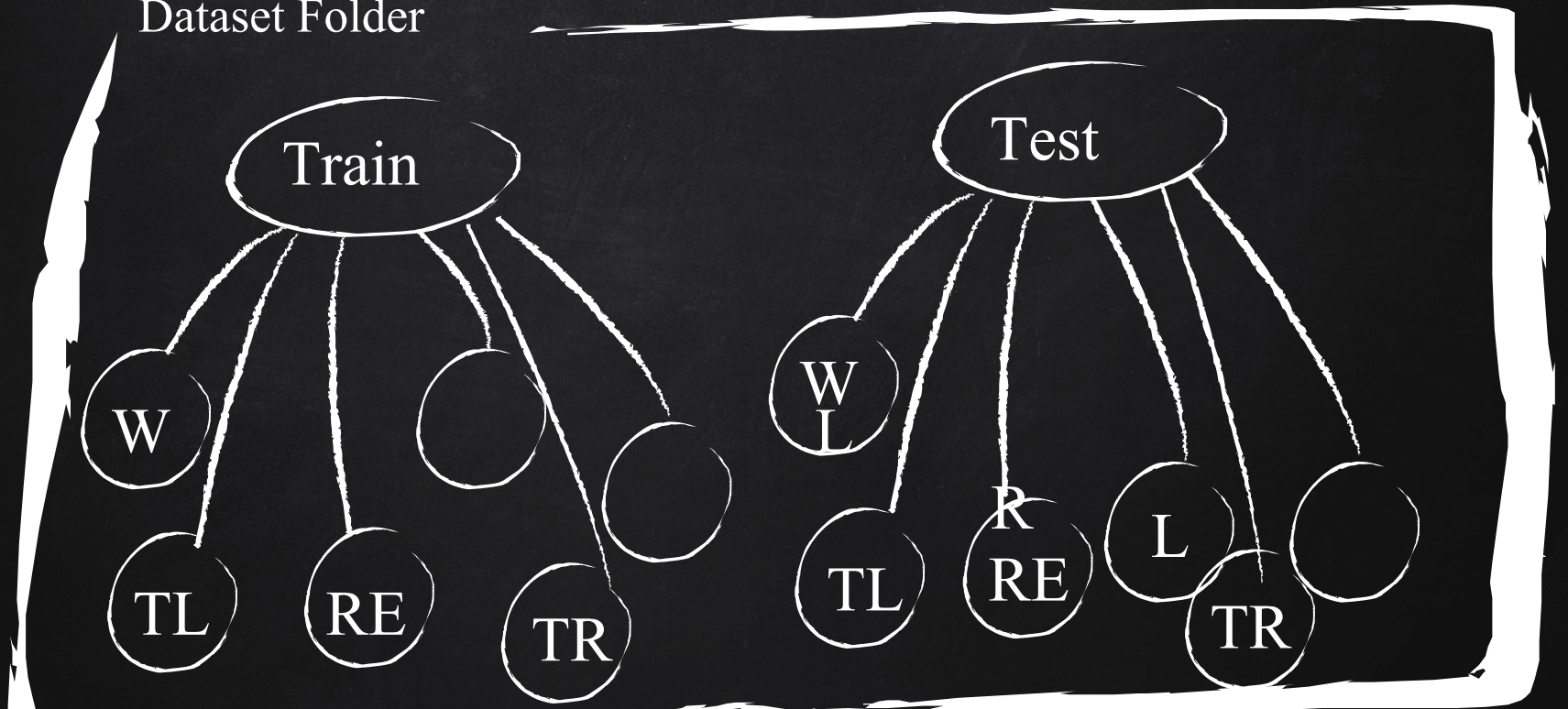
```python
import numpy as np
import cv2
import os

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 output classes
- .h5 file
  - Contains multidimensional arrays

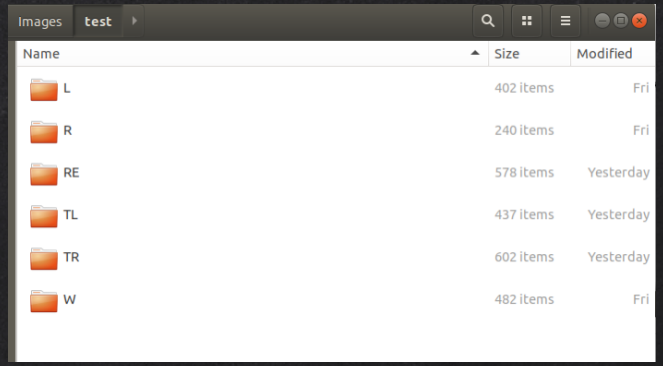
Final Dataset Size

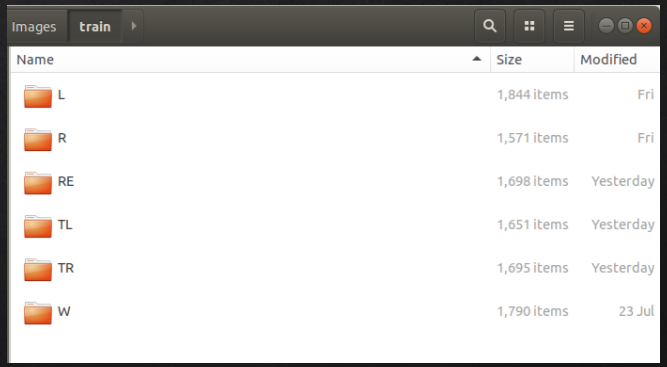
test

- W: 482
- L: 402
- R: 240
- RE: 578
- TL: 437
- TR: 602

train

- W: 1790
- L: 1917
- R: 1571
- RE: 1664
- TL: 1651
- TR: 1695





Code

- Initializes Camera
- Initializes GPU
- Writes image
- Reads Image
- Makes Prediction
- Sends Serial Command

File Edit View Search Terminal Help

```
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)
```

Model Issues: Dataset

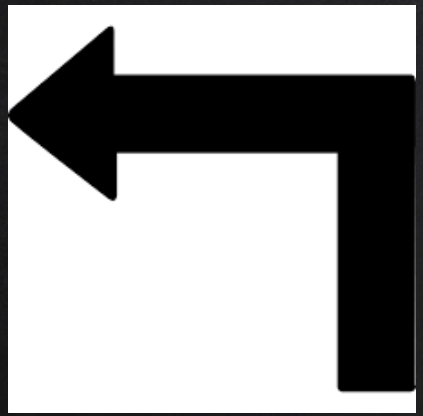
- Lack of sufficient image data
- Imbalance of output classes
  - The problem with continuous footage
  - Picking one guess output class
- Introduction of new output classes

test
- W: 482
- L: 402
- R: 240
- RE: 578
- TL: 437
- TR: 602

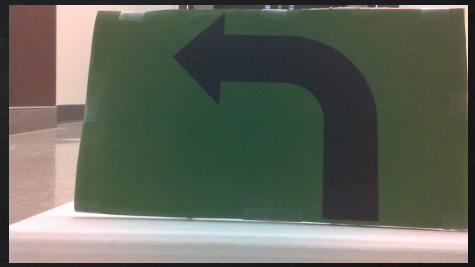
train
- W: 1790
- L: 1917
- R: 1571
- RE: 1664
- TL: 1651
- TR: 1695

Signs

Black and White



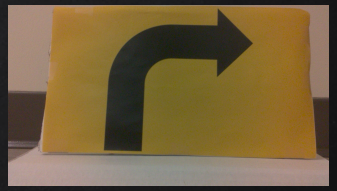
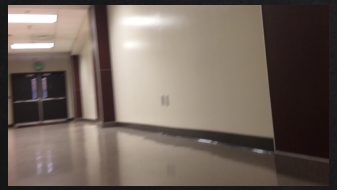
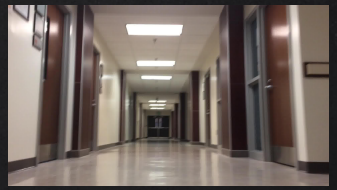
Color



Results

- Successfully train the car to self drive: correcting direction down the hall and reading signs for it to turn around, turn left, and turn right




File Edit View Search Terminal Help

Automatic suspend
Computer will suspend very soon because of inactivity.

```
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
```

display

File Edit View Search Terminal Help

Automatic suspend
Computer will suspend very soon because of inactivity.

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

display

(x=896, y=215) ~ R:104 G:94 B:91

File Edit View Search Terminal Help

Computer will suspend very soon because of inactivity.

```
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
```

display

(x=991, y=720) ~ R:50 G:69 B:64

File Edit View Search Terminal Help

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
```

display

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

File Edit View Search Terminal Help

Computer will suspend very soon because of inactivity.

```
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
```

display

(x=965, y=350) ~ R:145 G:62 B:59

```
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
```

display

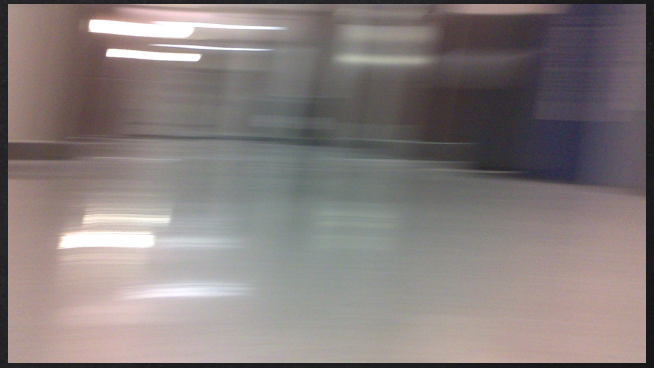
(x=965, y=350) ~ R:150 G:119 B:58

Demo Video

Error Factor: Out Of Focus / Blur

- Camera
  - Moving too fast
  - Shaking

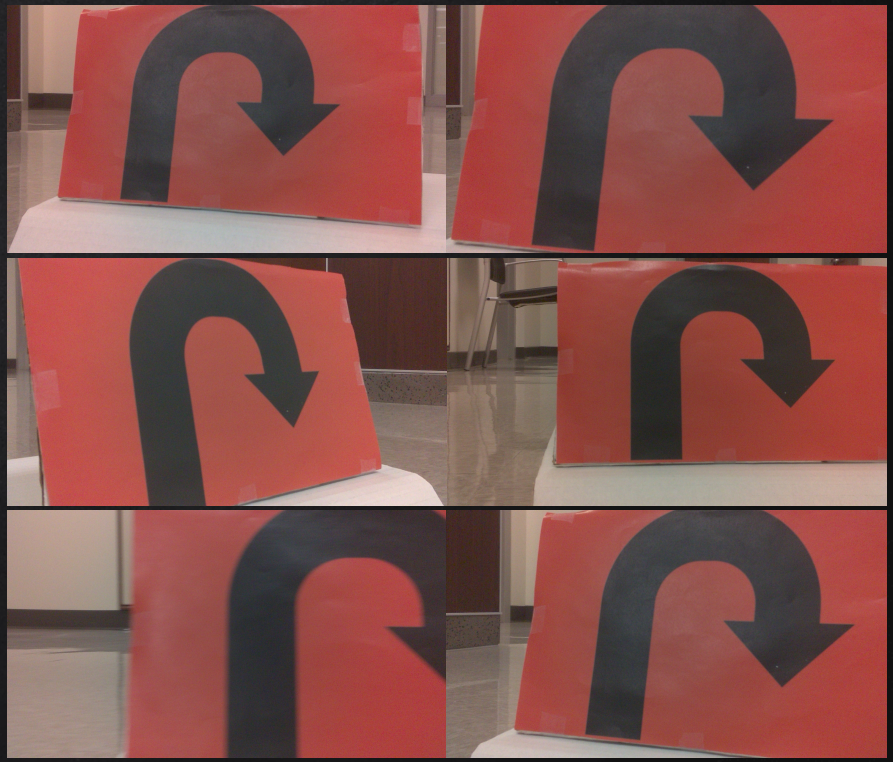
- No such examples in training
  - Random prediction



Error Factor: Out Of Focus / Blur

Error Factor: Field of View

- Need a Close Distance
  - ~ 1 - 1.5 Floor Tile
- Field of View (Rpi Camera v2)
  - 62.2 x 48.8 Degrees
- Missing the signs



Error Factor: Field of View

⚠ Limitation: Memory Size

```
GST_ARGUS: PowerService: requested_clock_Hz=16128
GST_ARGUS: Setup Complete, Starting captures for 0 seconds
GST_ARGUS: Starting repeat capture requests.
CONSUMER: Producer has connected; continuing.
2019-07-30 15:41:02.356957: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:931] ARM64 does not support NUMA - returning NUMA node zero
2019-07-30 15:41:02.357160: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: NVIDIA Tegra X1 major: 5 minor: 3 memoryClockRate(GHz): 0.9216
pciBusID: 0000:00:00.0
totalMemory: 3.87GiB freeMemory: 2.19GiB
2019-07-30 15:41:02.357250: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-07-30 15:41:04.632481: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-07-30 15:41:04.632560: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-07-30 15:41:04.632596: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2019-07-30 15:41:04.632801: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1634 MB memory) -> physical GPU (device: 0
, name: NVIDIA Tegra X1, pci bus id: 0000:00:00.0, compute capability: 5.3)
2019-07-30 15:41:34.032069: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.13GiB. The caller indicates that this is not a failure, but
may mean that there could be performance gains if more memory were available.
2019-07-30 15:41:34.760323: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.13GiB. The caller indicates that this is not a failure, but
may mean that there could be performance gains if more memory were available.
2019-07-30 15:41:35.759097: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.25GiB. The caller indicates that this is not a failure, but
may mean that there could be performance gains if more memory were available.
2019-07-30 15:41:35.852254: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.26GiB. The caller indicates that this is not a failure, but
may mean that there could be performance gains if more memory were available.
TR 99.35145378112793
```

Total RAM: 4GB

Free RAM: ~2GB:

- Image Capturing
- Put into the NN
  prediction outputs
- Overwrite the image file
- Generate the

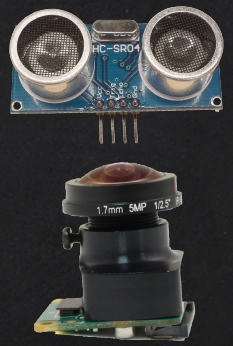
⚠ Limitation: Environment

- Diverse lighting
  - Different with training data

✘ Iconic landmark
✘ Clear boundaries / lines to guide the direction

Future Work

- Additional Sensors
  - Ultrasonic Distance Sensor
- Camera with Wide Angle Lens
  - ~110 degrees
- Positioning Device
  - Bluetooth device
  - Positional data



thanks!

Any questions?

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/