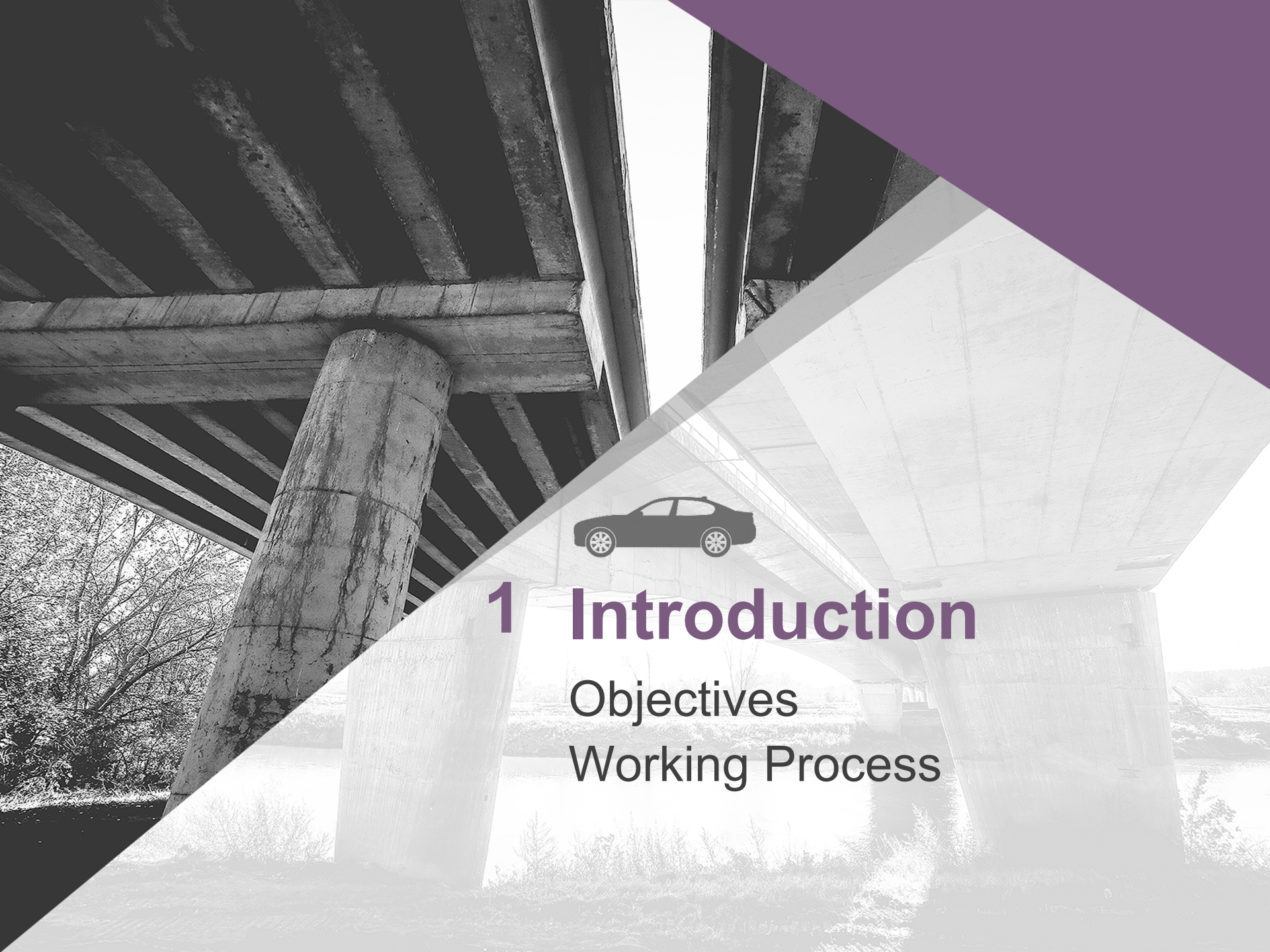
Mentor:
Dr Kwai Wong
Members:
Julian Halloy
Henry Lam
Beverly Chow

RC Autonomous Vehicle



1 Introduction

Objectives
Working Process

Objectives

Short term goal:

- Sign Recognition

Long term goals:

- Car-following
- Edge mesh

RC Autonomous Vehicle

Top layer

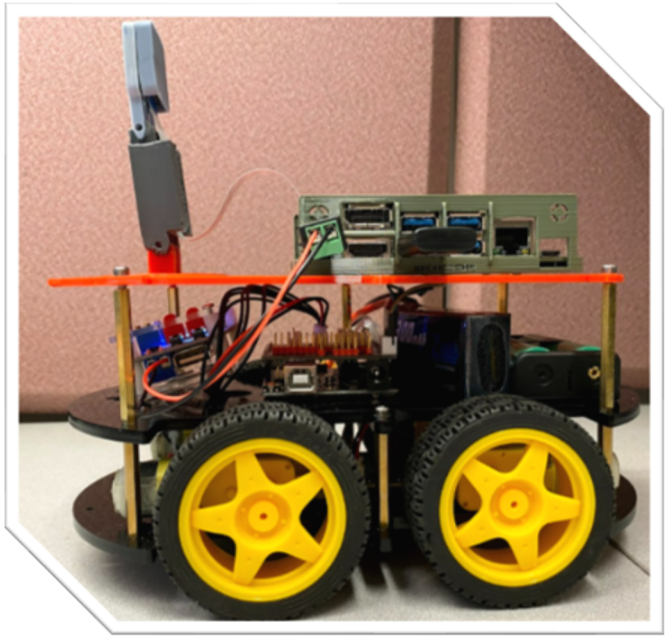
•Jetson Nano and camera

Upper layer

•UNO R3 and battery

Bottom layer

•DC motors and wheels



Jetson Nano and ELEGOO UNO R3

- Small single-board computer with high processing power

- Small processing device using microcontroller
- Provide control over external devices

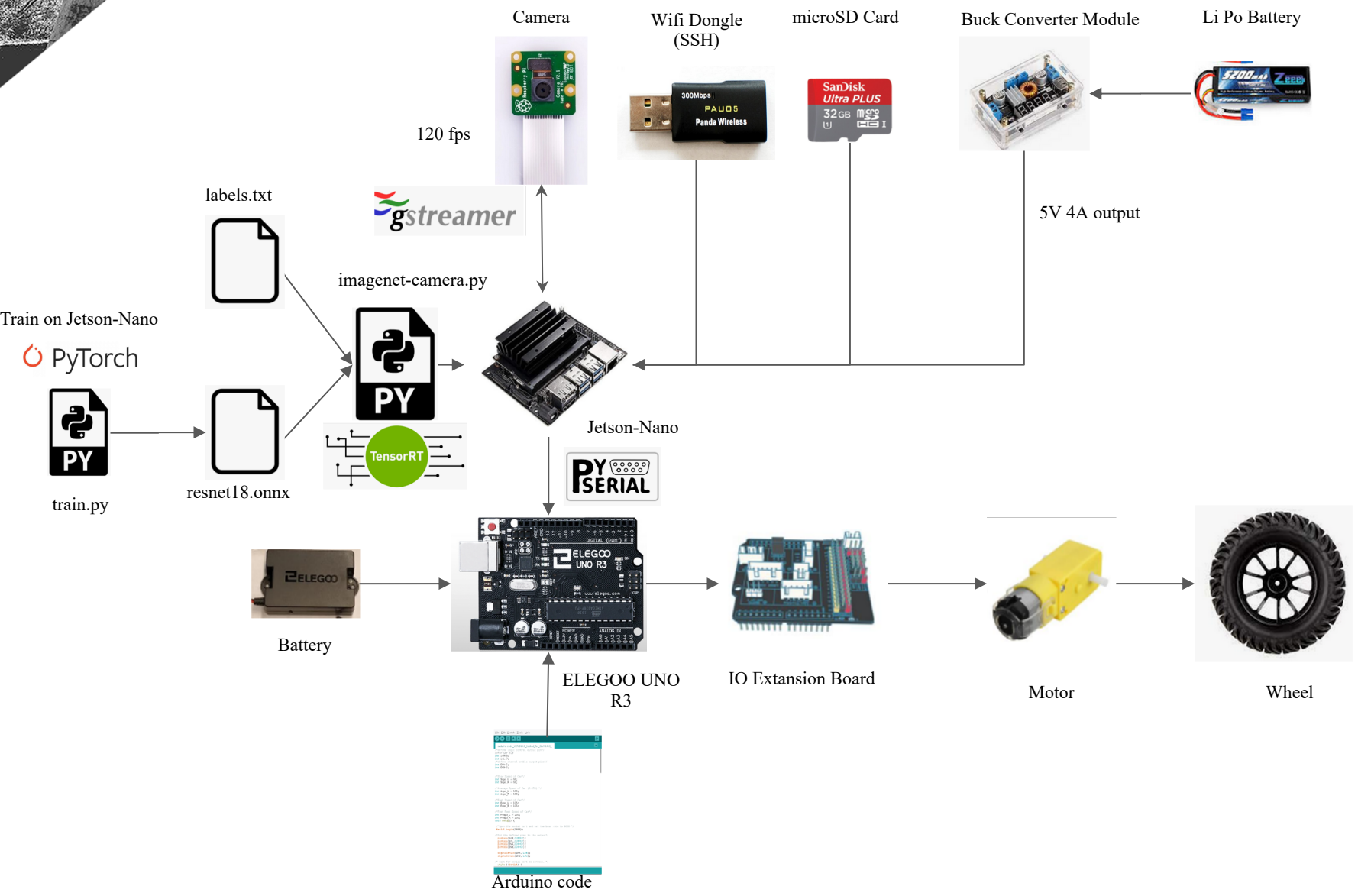
TensorRT

- C++ library that facilitates high-performance inference on NVIDIA GPUs
- Work with training frameworks
- Build features and applications based on new or existing deep learning models

Advantages :

1. Kernel Auto Tuning
2. Dynamic Tensor Memory Usage

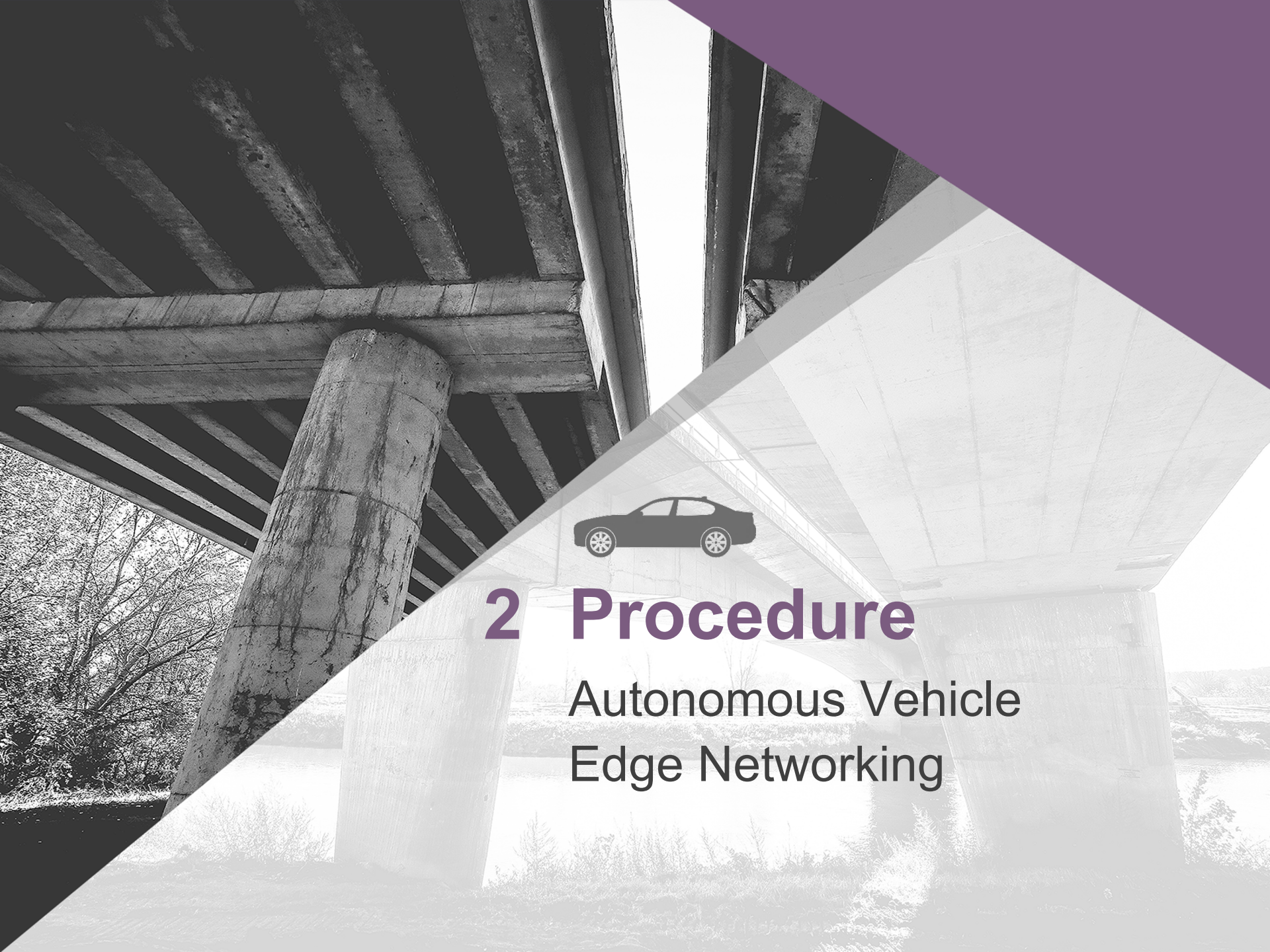
Schematic Diagram



Edge Networking

- edge device: a device that provide entry to network as a client, which can also execute and the command from the network.
- The major difference between the edge network and traditional network is the reliance on the server.
- There are different edge devices used in many industries like manufacturing, communication and transportation.

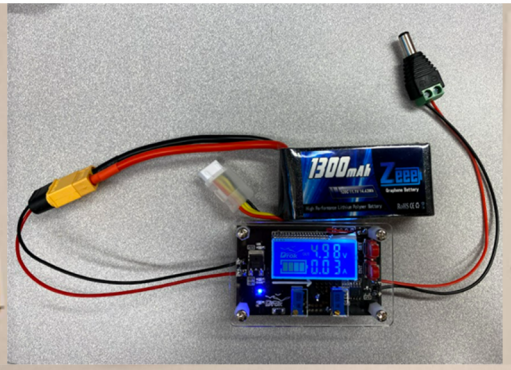
- Pros: Lower the reliance on 1 specified computing unit.
  Faster computational speed.
- Cons: Computational speed will depends on the numbers of edge devices in the networking

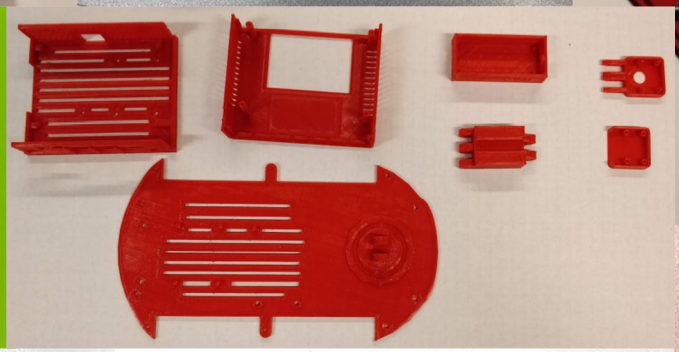


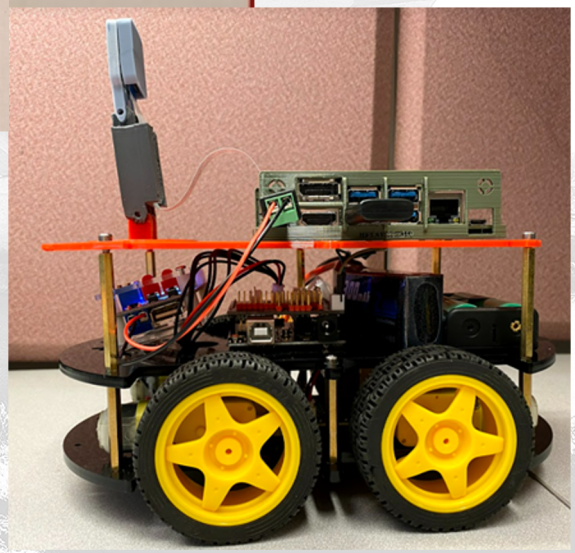
2 Procedure

Autonomous Vehicle Edge Networking

Autonomous Vehicle- Construct a Robot Car







Autonomous Vehicle- Set up Jetson Nano

- Write the unzipped Jetson Nano Developer Kit SD Card Image to the microSD Card
- Create Linux Account
- Install jetson-inference
- Install PySerial and gdown

Autonomous Vehicle- Communicate with Arduino

- “communicate-with-arduino.py”: give instructions to the UNO R3
- “Arduino-code.ino”: control the motors

Command	Action
w	Forward
s	Backwards
a	Left
d	Right
q	Stop
t	Return (Turn 180◦)
r	Left 90
y	Right 90
z	Slow Down
x	Speed Up

Autonomous Vehicle- Collect Data

- Run “communicate-with-arduino.py” and ”collect-img.py”
- W: The vehicle would go forward.
- L: The vehicle would turn left.
- R: The vehicle would turn right.
- TL: The photos that captured the “Turn Left Sign”
- TR: The photos that captured the “Turn Right Sign”
- RE: The photos that captured the “Return Sign”

Autonomous Vehicle- Model Training

Image

- Test
- Train
- Val

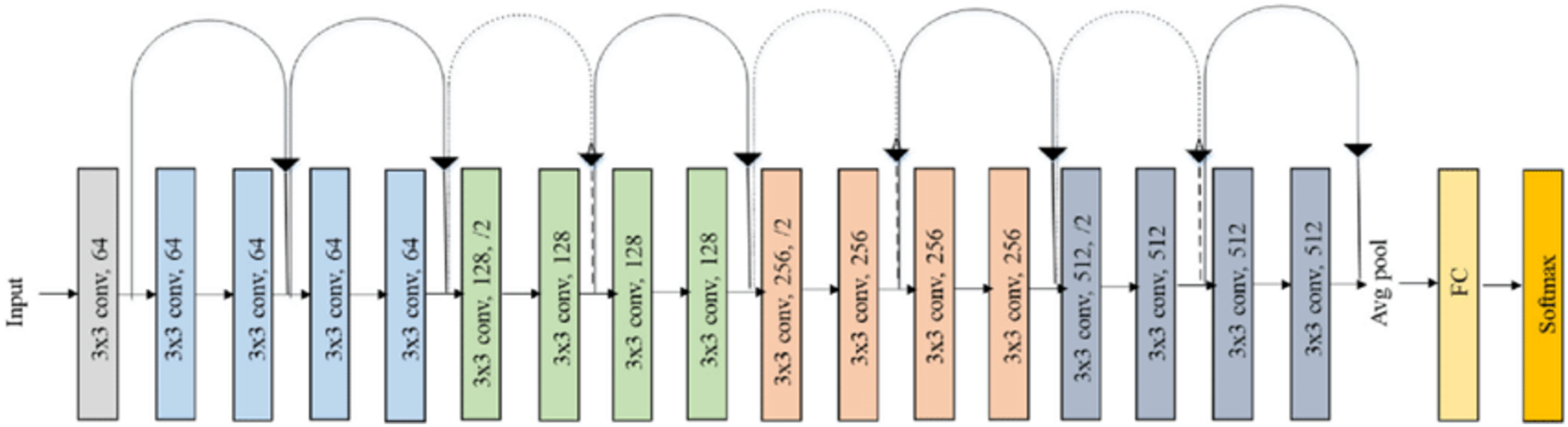
Class Label

- L (correct left)
- R (correct right)
- RE (Return)
- TL (Turn Left)
- TR (Turn Right)
- W (forward)

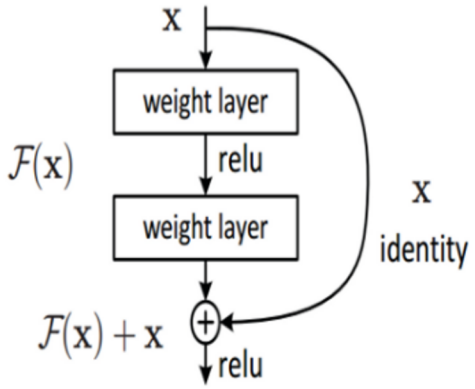
Run:

```
python3 train.py --model-dir=models data --epochs 10
```

ResNet18

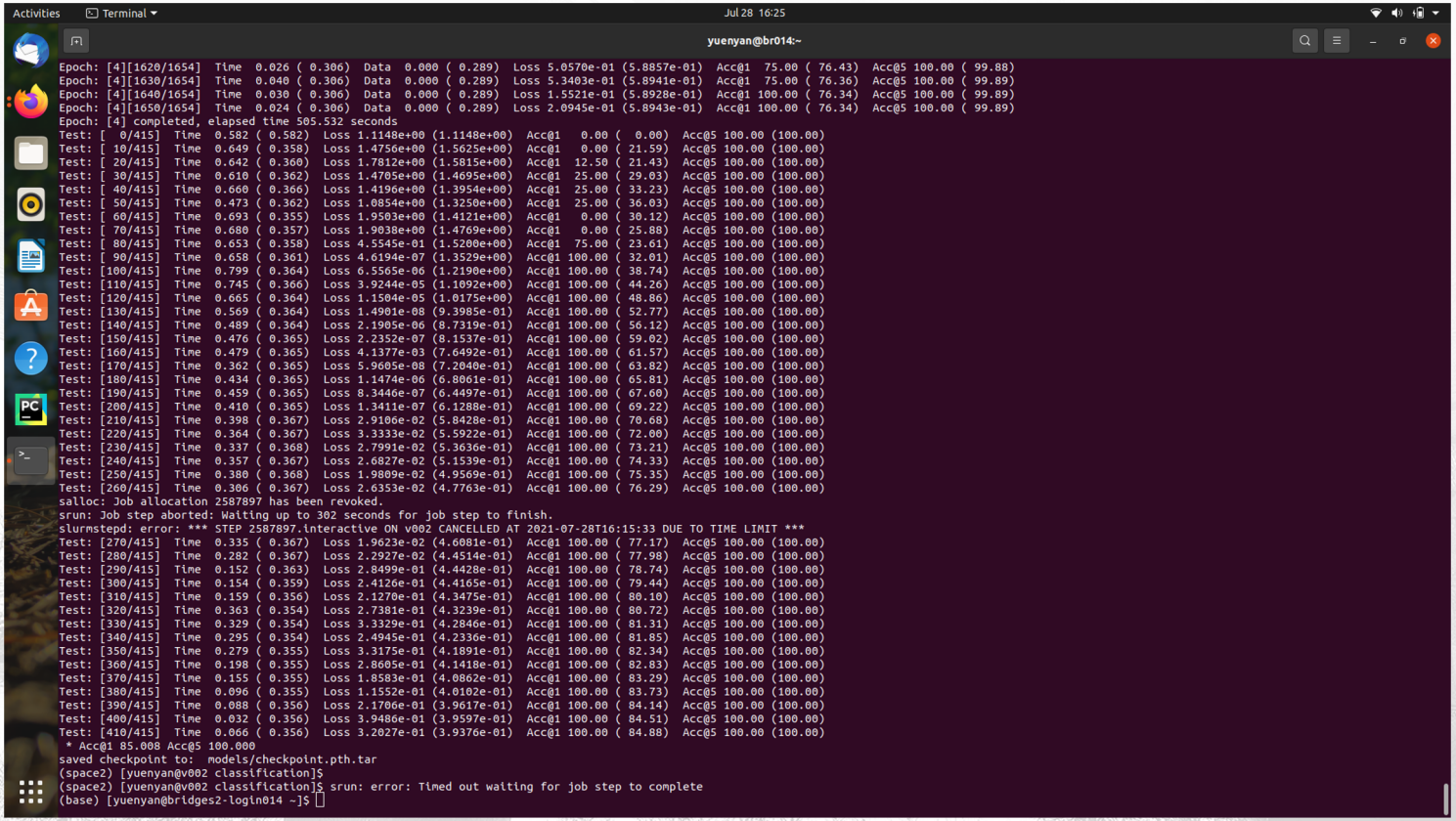


Advantage:
Prevent vanishing gradient



Autonomous Vehicle- Model Training

Result (Train on XSEDE) (5 epochs): Top1 average accuracy on testing set : 85.008



Autonomous Vehicle- Sign Recognition

```
# load the recognition network
net = jetson.inference.imageNet(opt.network, sys.argv)

# create video sources & outputs
input = jetson.utils.videoSource(opt.input_URI, argv=sys.argv)
output = jetson.utils.videoOutput(opt.output_URI, argv=sys.argv + is_headless)
font = jetson.utils.cudaFont()

with serial.Serial('/dev/ttyACM0', 9600, timeout=10) as ser:
    # process frames until the user exits
    while True:
        # capture the next image
        img = input.Capture()

        # classify the image
        class_id, confidence = net.Classify(img)
        print(class_id)
        print(confidence)

        if (confidence > 0.3):
            if (class_id == 5):
                ser.write(bytes('W\n', 'utf-8'))
                turn = 0
                print('forward')

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

            if (class_id == 1):
                ser.write(bytes('D\n', 'utf-8'))
                turn = 0
                print('right')

            if (class_id == 2):
                ser.write(bytes('R\n', 'utf-8'))
                print('turn left')
            # ser.write(bytes('W\n','utf-8'))
            # time.sleep(0.5)

            if (class_id == 3):
                ser.write(bytes('Y\n', 'utf-8'))
                print('turn right')
                turn = 0

            if (class_id == 4):
                if turn == 0:
                    turn = 1
                    ser.write(bytes('T\n', 'utf-8'))
                    print('Return')
                else:
                    ser.write(bytes('W\n', 'utf-8'))

        # find the object description
        class_desc = net.GetClassDesc(class_id)

        # overlay the result on the image
        font.OverlayText(img, img.width, img.height, "{:05.2f}% {:s}".format(confidence * 100, class_desc), 5, 5,
                         font.White, font.Gray40)

        # render the image
        output.Render(img)

        # update the title bar
        output.SetStatus("{:s} | Network {:.0f} FPS".format(net.GetNetworkName(), net.GetNetworkFPS()))

        # print out performance info
        net.PrintProfilerTimes()

        # press 'Q' to stop
        if keyboard.is_pressed('q'):
            ser.write(bytes('Q\n', 'utf-8'))
            print('Stop')
            break
        # exit on input/output EOS
        if not input.IsStreaming() or not output.IsStreaming():
            break
```

Autonomous Vehicle- Sign Recognition

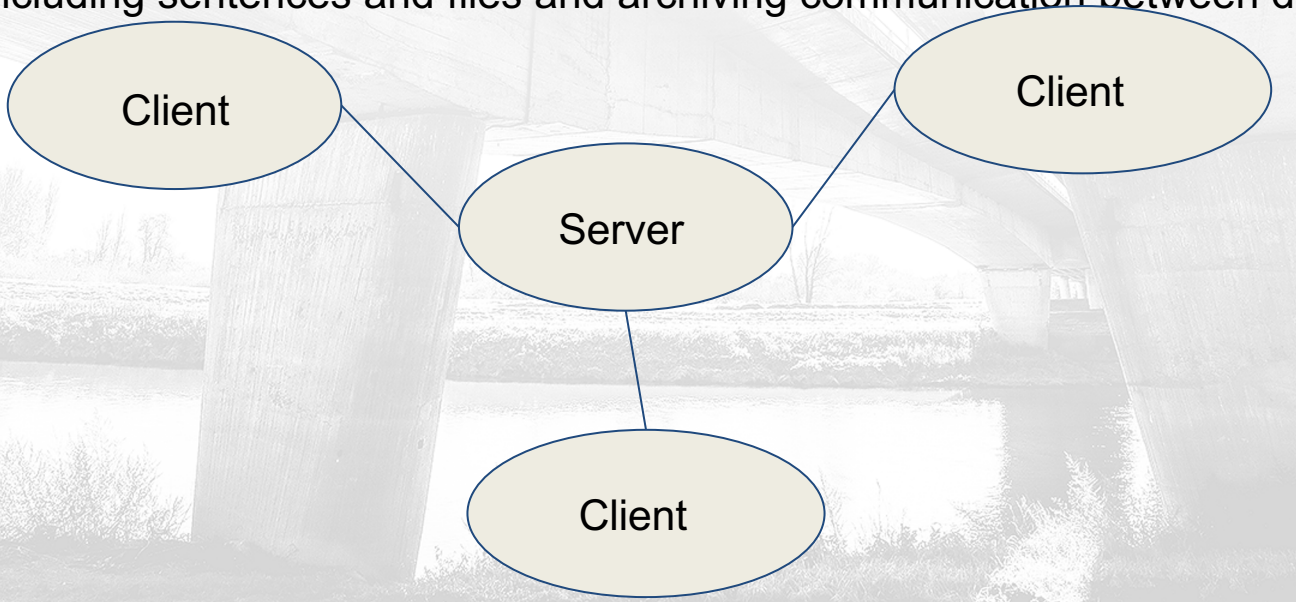
python3 imagenet-camera-c.py --model=/home/car4/jetson-inference/python/training/classification/models/resnet18.onnx --labels=/home/car4/jetson-inference/python/training/classification/data/labels.txt --input_blob=input_0 --output_blob=output_0 csi://0

```
class 0000 - 0.157939  (L)
class 0001 - 0.145027  (R)
class 0002 - 0.013725  (RE)
class 0003 - 0.487710  (TL)
class 0004 - 0.038978  (TR)
class 0005 - 0.156621  (W)
imagenet:  48.77104% class #3 (TL)

[TRT]    ------------------------------------------------
[TRT]    Timing Report /home/car4/Jetson-Nano/jetson-inference/python/training/classification/models/resnet18.onnx
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.05271ms  CUDA   0.66880ms
[TRT]    Network       CPU  30.08979ms  CUDA  24.52917ms
[TRT]    Post-Process  CPU   0.09589ms  CUDA   0.09354ms
[TRT]    Total         CPU  30.23839ms  CUDA  25.29151ms
[TRT]    ------------------------------------------------

class 0000 - 0.106094  (L)
class 0001 - 0.099528  (R)
class 0002 - 0.015095  (RE)
class 0003 - 0.641093  (TL)
class 0004 - 0.052403  (TR)
class 0005 - 0.085787  (W)
imagenet:  64.10929% class #3 (TL)

[TRT]    ------------------------------------------------
[TRT]    Timing Report /home/car4/Jetson-Nano/jetson-inference/python/training/classification/models/resnet18.onnx
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.05417ms  CUDA   0.65958ms
[TRT]    Network       CPU  28.44060ms  CUDA  22.65208ms
[TRT]    Post-Process  CPU   0.22677ms  CUDA   0.22568ms
[TRT]    Total         CPU  28.72154ms  CUDA  23.53735ms
[TRT]    ------------------------------------------------

class 0000 - 0.073603  (L)
class 0001 - 0.072188  (R)
class 0002 - 0.018280  (RE)
class 0003 - 0.694449  (TL)
class 0004 - 0.087400  (TR)
class 0005 - 0.054079  (W)
imagenet:  69.44489% class #3 (TL)
```

Edge Networking-Socket

For developing the communication between edge devices, we used socket module in python language, which is a module for communicating through defining server and client machines by socket endpoints.

We first set one vehicle as the host of server and other vehicles will connect to it as clients. By inserting socket module, we can set up the pipeline for different message types including sentences and files and archiving communication between devices.



3 Result

Result
Analysis

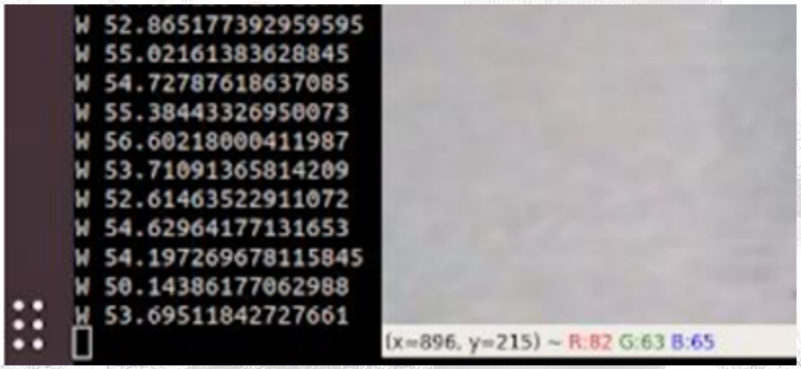
Result- Sign Recognition (ImageAI)

- Wrong classification outcome

```
This message will be only logged once.
result on testing set:
TL   :  100.0
W  :  0.0
TR   :  0.0
RE   :  0.0
R  :  0.0
L  :  0.0
result on training set:
TL   :  100.0
W  :  0.0
TR   :  0.0
RE   :  0.0
R  :  0.0
L  :  0.0
(space1) [yuenyan@v002 yuenyan]$ timed out wai
salloc: Relinquishing job allocation 2314846
(base) [yuenyan@bridges2-login011 ~]$
```

The Final Report written in 2019:

- Most of the accuracies: 70-80%
- Lowest accuracy: 52%



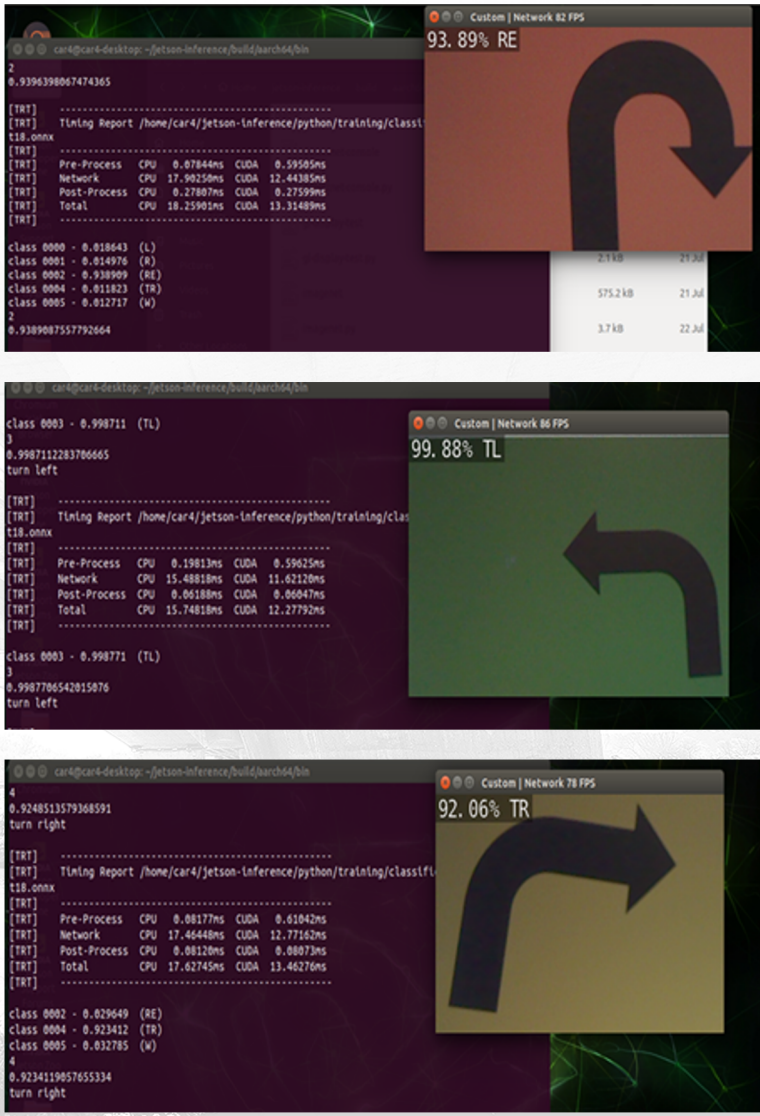
Result with 50% accuracy

Result with 80% accuracy

Result- Sign Recognition (TensorRT)

- The accuracy of classification results can always greater than 90%.

  Problems:
- SSH: Program cannot be stopped by inputting commands.



Analysis-image capturing

The possible reasons of improvement:

•TensorRT is designed for Nvidia GPU product.

-The previous team mentioned that great portion of Jetson-Nano Memory is used when classification is processing. Consider that Jetson-Nano only have 2 GB of internal memory, lack of memory is possible to affect the classification result.

-TensorRT has already been optimized for Jetson-Nano to used, which means under normal circumstances, when Jetson-Nano is doing classification, memory will not be running out.

•The data sets have been optimized.

-Previous team also mentioned that, in their training stage they cannot finished the sorting of images for all output classes, when they finished the sorting, they already have had no time to train a new model.

-Since this time, we ultimately used their completed data sets for our model training, It is possible that our model have a higher accuracy than theirs.

Analysis-image capturing

The possible reasons of improvement:

•The database size difference between imageAI and Imagenet.

-In terms of size of data set, imagenet is the biggest among its kind, it has 14 millions individual pictures for 21000 classes and groups, which means it can provide a better reference for our model training process than imageAI.

-Because imageAI is a relatively small production that develop by a only 2 members research team, they probably do not have such a great database as Imagenet.

The reason of the program error

•It probably related to display window

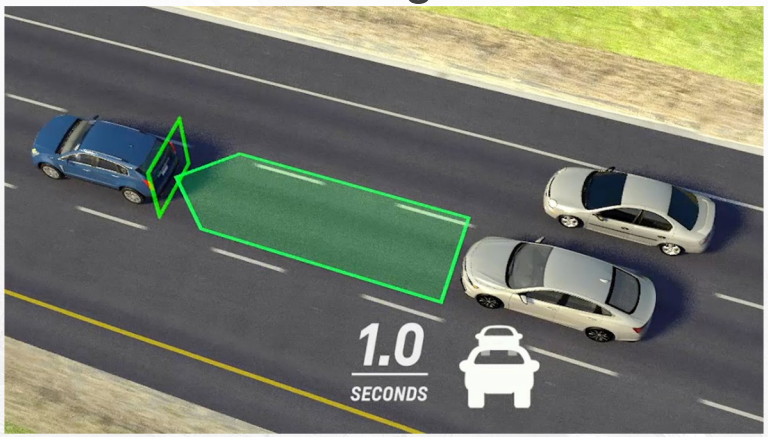
-When we tried to work on imageAI, we have the similar problem that the function cannot be stopped, we discussed with our colleague Julian from UTK, we found that it probably related to that display window cannot be opened on the SSH client. Then we thought this is a possible reason.

4 Progressing Research

Work
Application

Progressing research works: Car following

After the vehicle can drive autonomously, our next goal is connect more vehicles to follow the heading one to become a fleet.



The concept and procedure is similar to the autonomous feature,

- take many photos
- sought photos
- train the model
- classification and turn it signals to arduinos

The Only difference: The signal source become the heading car instead of sign board

Progressing research work: Edge Networking

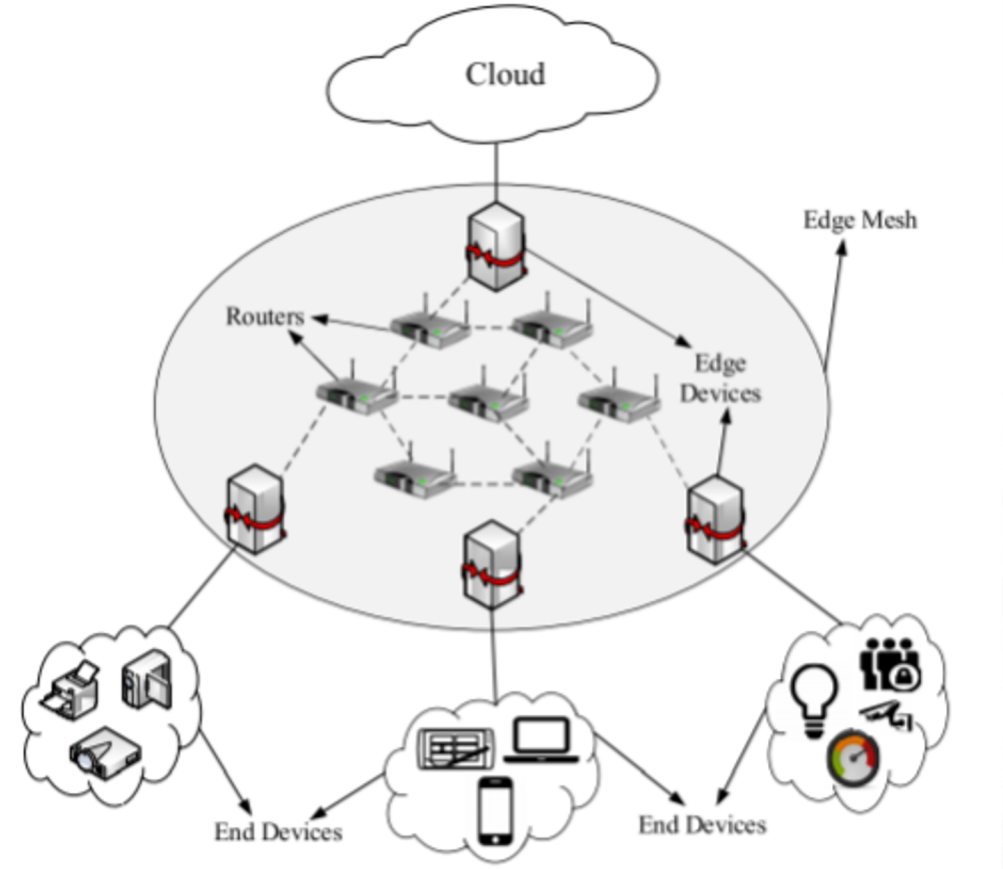
Other than Sign recognition and Car following, we have also been working on Communication for cars to exchange information.

Concept: The RC Autonomous Vehicle = edge device.

In this idea, edge devices are not only executors, but also computing units. In the Edge network, edge devices will do the major decisions, which is different from the traditional network that use a centralized cloud to compute Sahni(Yuvraj, Jiannong Cao, Shigeng Zhang, & Lei Yang, 2017).

1.Sahni, Yuvraj, Jiannong Cao, Shigeng Zhang, & Lei Yang. (2017). Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things. *IEEE Access*, *5*, 16441–16458. https://doi.org/10.1109/ACCESS.2017.2739804

Progressing research work: Edge Networking



- Edge devices cluster is formed,
- the jobs will split to different parts
- parallel computing
- Similar process to a **Super Computer.**

5 Conclusion

Conclusion

- TensorRT resolved the major problem of Jetson-Nano's memory limitation which also improve the Autonomous features performance compare with the old set of software.

- Jetson-Nano with TensorRT may have greater possibility to access some advance research work which provide greater space for future research on this project.


THANK YOU