بسم الله الرحمن الرحيم

AUTO PARKING OF VEHICLE BY USING SUPERVISED LEARNING APPROACH

SUPERVISOR: DR. MUWAHIDA LIAQUAT

GROUP MEMBERS:
MAZAHIR HUSSAIN
MUSTAQEEM ASHRAF
SHAMSA KANWAL
TOOBA ANWAR

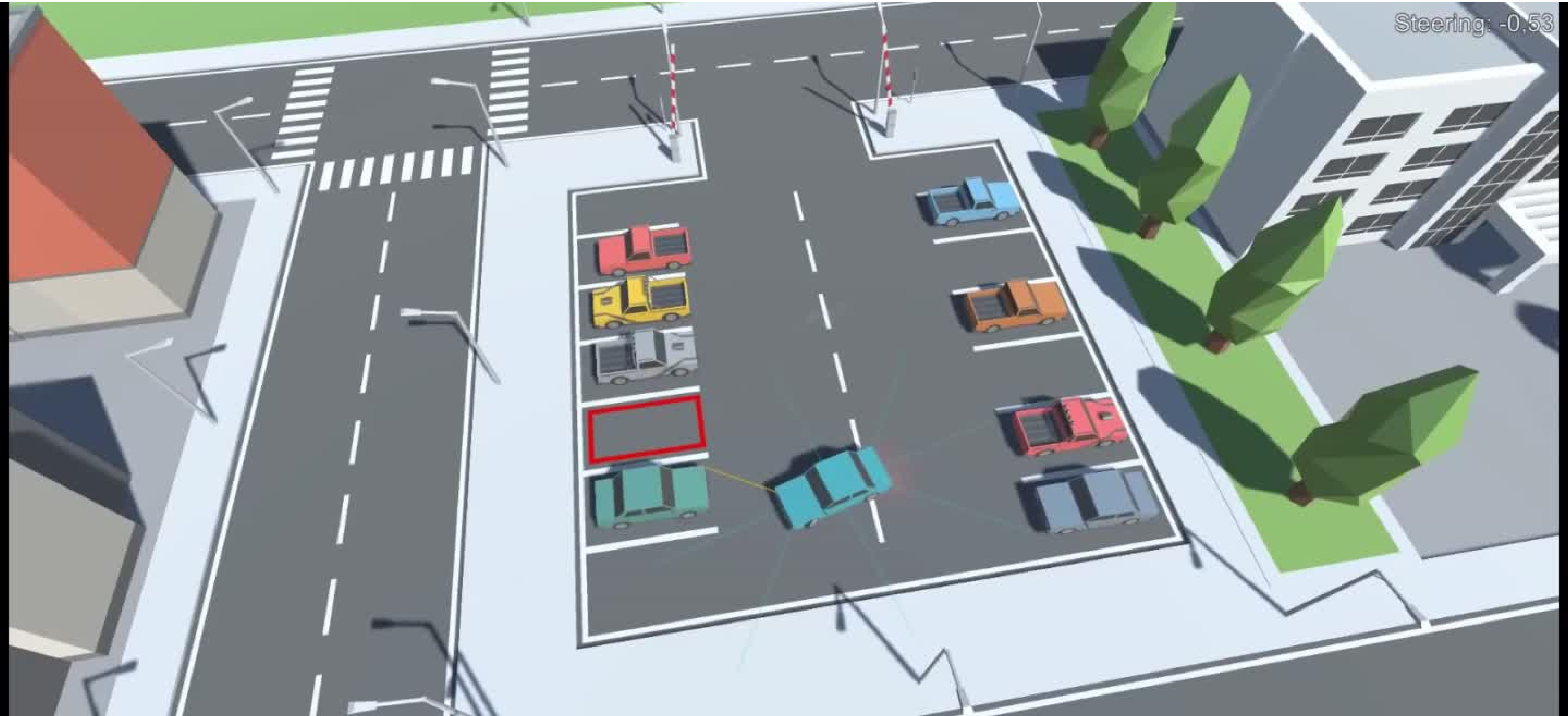
Introduction

1. Autonomous vehicles are designed to perform the tasks done by human drivers.
2. Its main purpose is to reduce man power and shift it to machines.
3. Our projects starts when the car enter in parking area
4. A CNN is trained which takes pictures from a single facing camera as an input.
5. CNN model detects the empty slot and move the car toward the parking point.

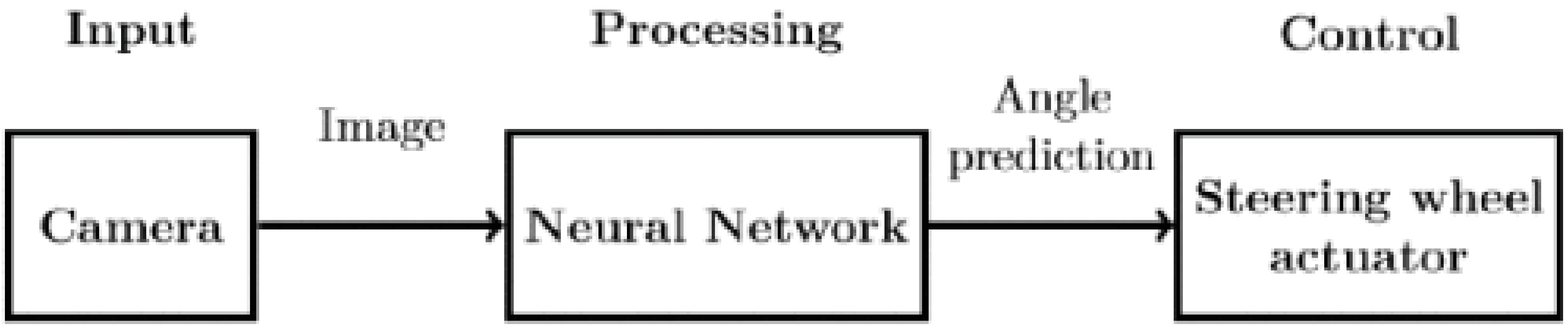
OBJECTIVES

1. MERCEDES company introduced car parking assistant in its mercedez-benz to assist the driver for parking
2. BMW also done similar work as mercedez their project name is BMW PARK ASSISTANT
3. Both companies worked on asistant parking in which car is still being parked by a human driver
4. Objective of our project is to park the vehicle autonomously by using camera sensor without the interference of human driver

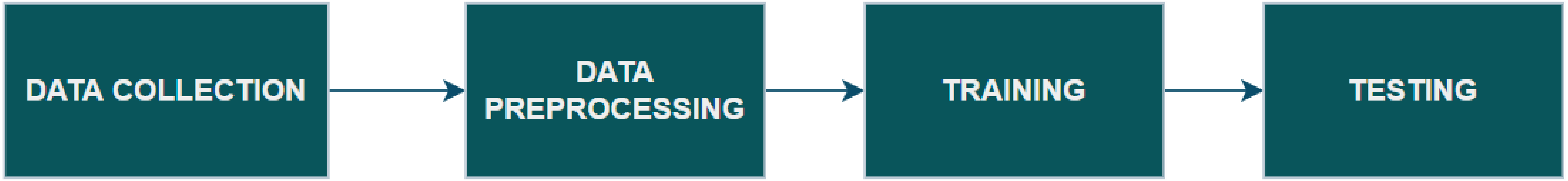
Objective



Block Diagram



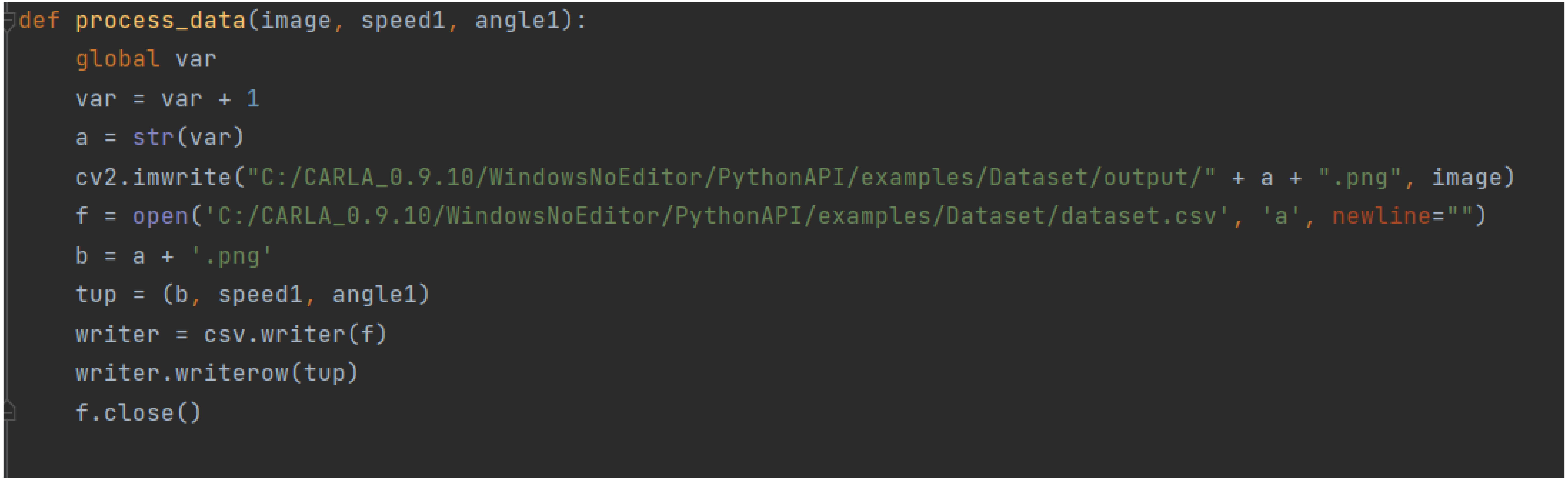
FLOW CHART



Data Collection

- CARLA
- Camera sensor
- Collection of Raw Images
- Labeling(Angle, Throttle)

Data Collection



```
def process_data(image, speed1, angle1):
    global var
    var = var + 1
    a = str(var)
    cv2.imwrite("C:/CARLA_0.9.10/WindowsNoEditor/PythonAPI/examples/Dataset/output/" + a + ".png", image)
    f = open('C:/CARLA_0.9.10/WindowsNoEditor/PythonAPI/examples/Dataset/dataset.csv', 'a', newline="")
    b = a + '.png'
    tup = (b, speed1, angle1)
    writer = csv.writer(f)
    writer.writerow(tup)
    f.close()
```

Data Preprocessing

- Image resize

  1280x640 -> 320x160

  Area Interpolation

- Image Normalization

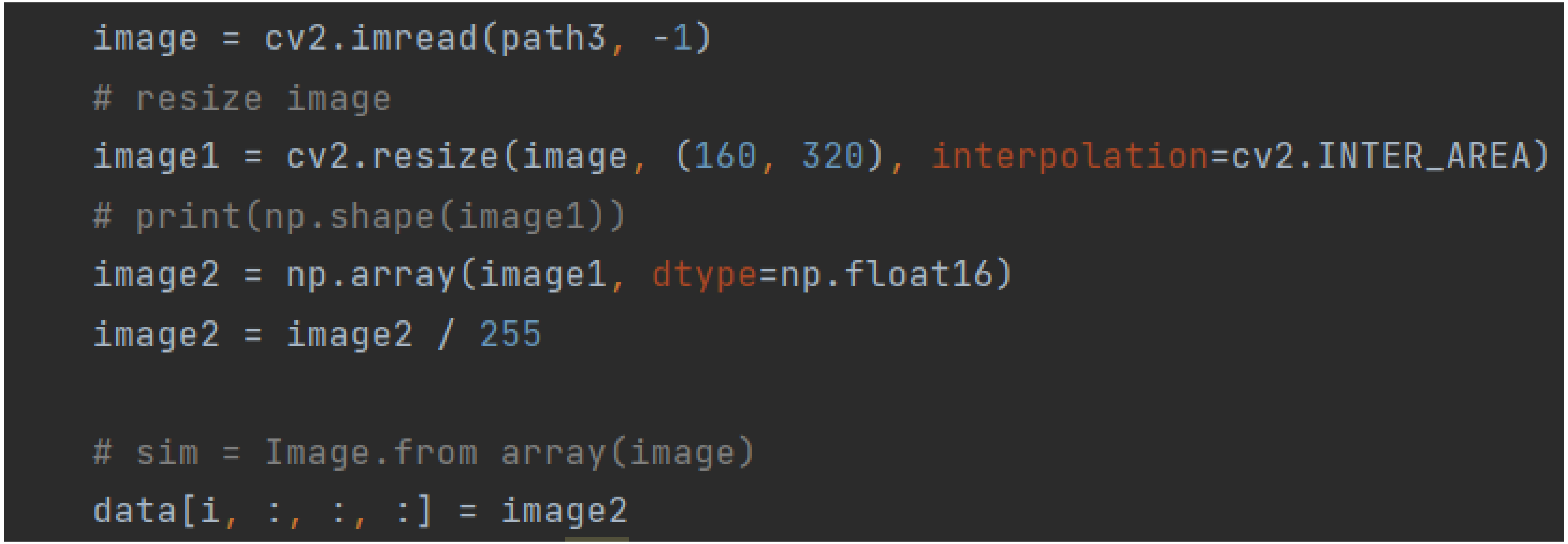
  int -> float16

- Batch size

  For GPU = 1024

  CNN Batch size = 32

Data Preprocessing



```
image = cv2.imread(path3, -1)
# resize image
image1 = cv2.resize(image, (160, 320), interpolation=cv2.INTER_AREA)
# print(np.shape(image1))
image2 = np.array(image1, dtype=np.float16)
image2 = image2 / 255

# sim = Image.from array(image)
data[i, :, :, :] = image2
```

Training

- Splitting of data
- CNN Model

  6 Conv. layers

  Dropout(50%)

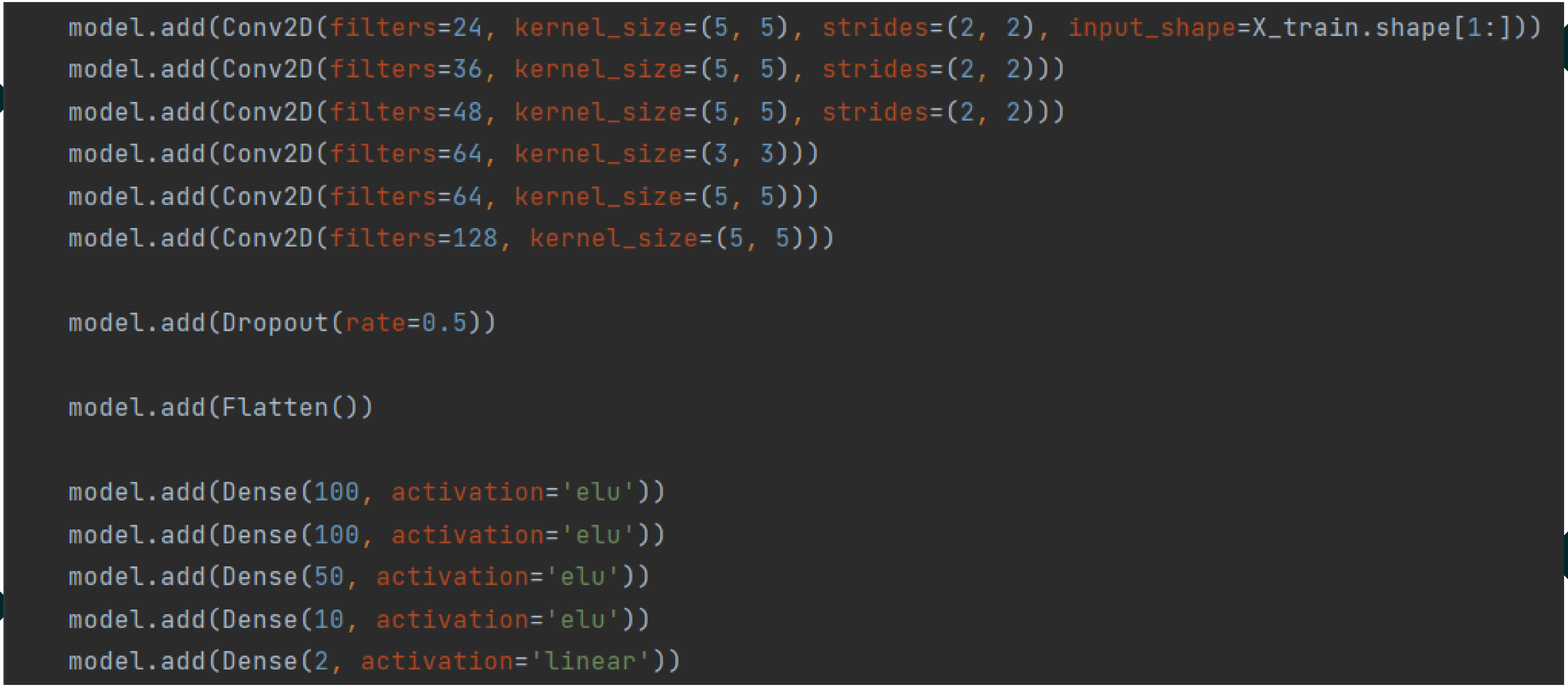
  5 Dense layers

  Activation Function
- Hyper parameter

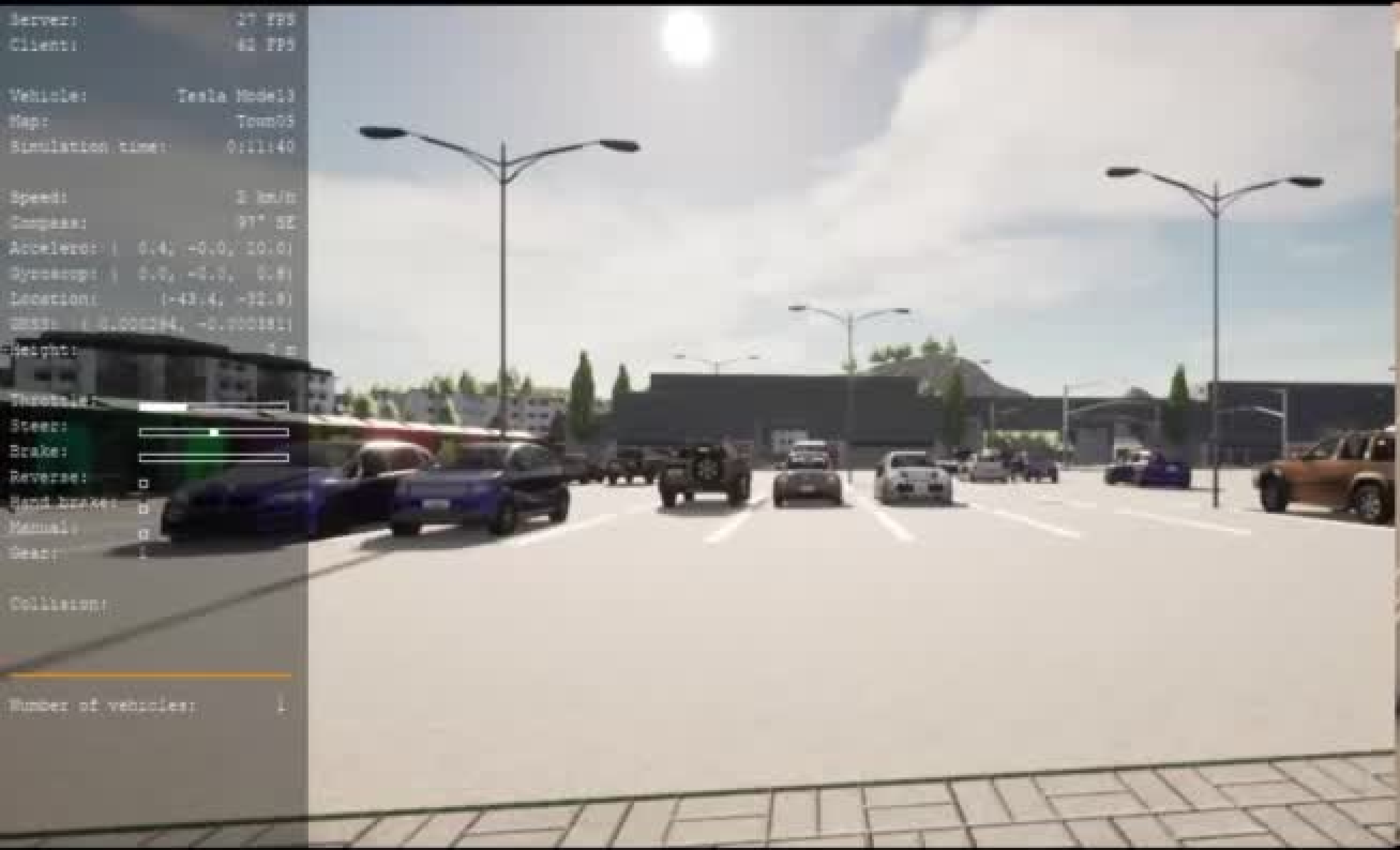
  Epoch = 50

  Learning rate = 0.001

Training

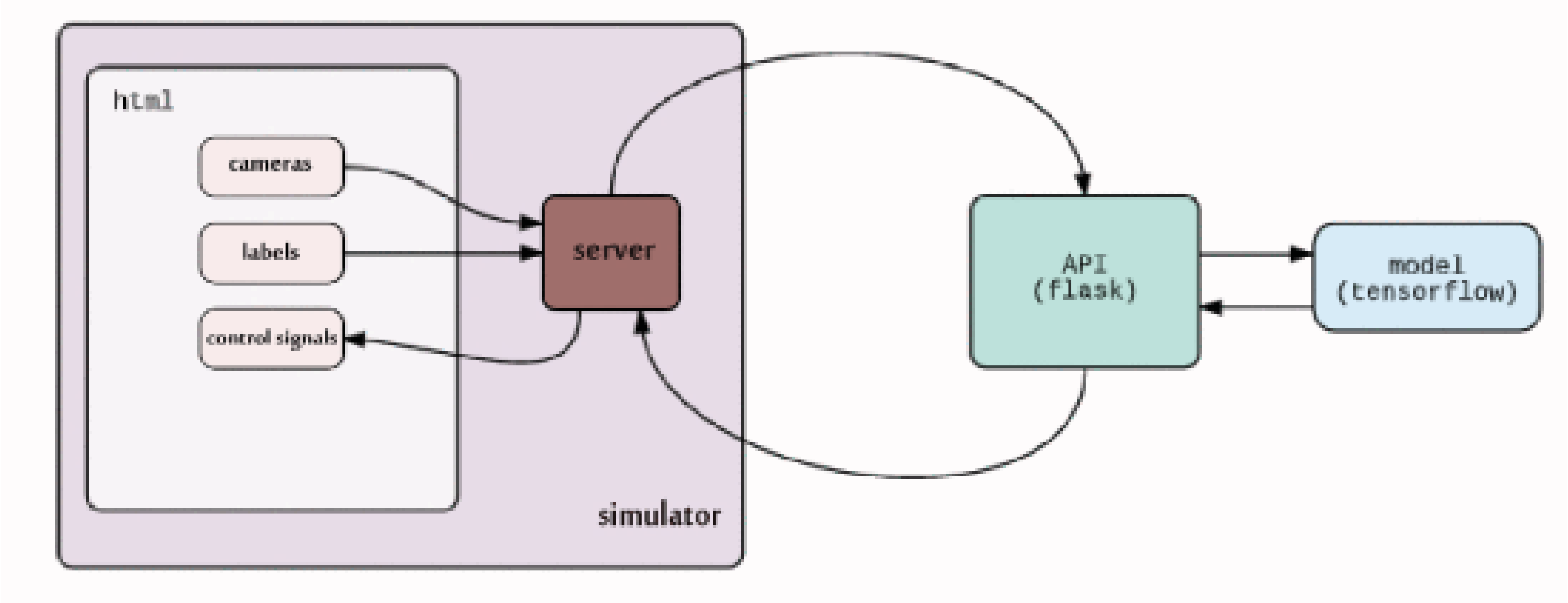


```
model.add(Conv2D(filters=24, kernel_size=(5, 5), strides=(2, 2), input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=36, kernel_size=(5, 5), strides=(2, 2)))
model.add(Conv2D(filters=48, kernel_size=(5, 5), strides=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3)))
model.add(Conv2D(filters=64, kernel_size=(5, 5)))
model.add(Conv2D(filters=128, kernel_size=(5, 5)))

model.add(Dropout(rate=0.5))

model.add(Flatten())

model.add(Dense(100, activation='elu'))
model.add(Dense(100, activation='elu'))
model.add(Dense(50, activation='elu'))
model.add(Dense(10, activation='elu'))
model.add(Dense(2, activation='linear'))
```



```
Server:                    27 FPS
Client:                    62 FPS

Vehicle:             Tesla Model3
Map:                       Town05
Simulation time:          0:11:40

Speed:                     2 km/h
Compass:                   97° SE
Accelero: ( 0.4, -0.0, 10.0)
Gyroscop: ( 0.0, -0.0,  0.8)
Location:        (-43.4, -32.8)
GNSS:   ( 0.000294, -0.000381)
Height:                       0 m

Throttle:
Steer:
Brake:
Reverse:      □
Hand brake:   □
Manual:       □
Gear:         1

Collision:

Number of vehicles:           1
```

Software implemention



Technologies/Software Used

- CARLA
- TensorFlow 2.8.0/2.9.1 with Keras as backend.
- To train our network on Graphical Processing Unit (GPU), we used cuda framework and drivers.
- Language: Python 3.10 & 3.7 to integrate with CARLA
- Libraries: TensorFlow, Open CV, Sklearn

Machine For Training

- Processor: Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz
- RAM 16GB
- GPU: NVIDIA GeForce GTX 1650 4GB
- Computational power = 7.5

Results

Results

- In Carla Simulator we used 3 different angles to park the car
- Which is 0° ,45° ,-45°
- As shown in the video car is being parked at all the above angles which is our desired result

DRAWBACKS

- Without Graphical Processing Unit (GPU) processing the training would take an enormous amount of time. This means that the process of developing, testing and running the DNN was very GPU dependent.
- To increase the accuracy, high computational power is required.
- Driving speed must be below 13 km/h

DRAWBACKS

- Auto-parking may not work with textured road surfaces such as cobblestone or brick.
- Auto-parking performance depends on the ability of the cameras (as we are using single camera) to determine the vehicle's proximity to curbs, objects, and other vehicles
- While raining droplets on camera may effect the result of camera

Conclusions

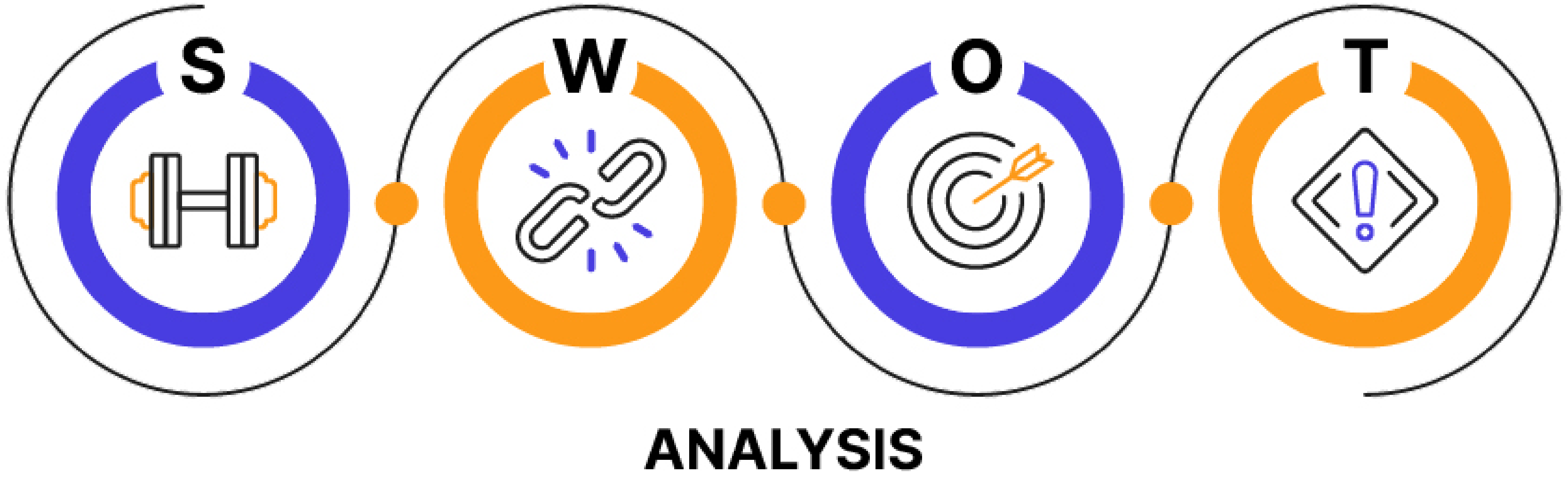
- The main motive for this work is to park the vehicle autonomously by using camera sensor without the interference of human driver.
- we used supervised learning as the approach for training the model.
- The neural networks used in this project is a regression one to predict the output of the steering angle and throttle.
- we have tackled the problem of perpendicular parking by using Deep Convolutional Neural Networks (CNNs).

Conclusions

- Collected Data from CARLA SIMULATOR by using camera sensor
- Our model is inspired by work done on autonomous vehicles and self-driving cars of NVIDIA.
- Testing proves the high reliability and efficiency of proposed method.
- The model was able to detect useful road features on its own, i.e., with only the human steering angle and throttle as labels.

SWOT Analysis

- **Strengths**: End-to-End Deep Learning approach.
- **Weaknesses**: A lot of data required for training
- **Opportunities**: Using more sensors like LIDAR, RADAR and ultrasonic sensors accuracy can further be increased.
- **Threats**: no threats so far.



Suggestions for Future Work

- Dataset will be available for those who wish to work on project of same nature in the future.
- Improve the car navigation accuracy by adding more functionalities to the model such as avoiding encountered obstacles on the track.
- The approach can be applied to self-driving cars with the same localization system or it can be replaced with GPS and high-definition (HD) maps.

Suggestions for Future Work

- Other parking techniques such as parking in the presence of static and dynamic obstacles, e.g., Pedestrians, other vehicles, etc. can be achieved.
- Using more sensors like LIDAR, RADAR and ultrasonic sensors accuracy can further be increased.
- Same model can be used to improve the accuracy by adding more convolutional layers.
- Implementing the CNN on a physical car could have a great impact on the accuracy of the model.

Q & A SESSION

References

- **[1]** Lin YL, Li L, Dai XY, Zheng NN, Wang FY. **Master general parking skill via deep learning.** IEEE Intell Veh Symp Proc 2017:941–6. https://doi.org/10.1109/IVS.2017.7995836.
- **[2]** Gamal O, Imran M, Roth H, Wahrburg J. **Assistive Parking Systems Knowledge Transfer to End-To-End Deep Learning for Autonomous Parking**. 2020 6th Int Conf Mechatronics Robot Eng ICMRE 2020 2020:216–21. https://doi.org/10.1109/ICMRE49073.2020.9065014.
- **[3]**Convolutional Neural Network for a Self-Driving Car in a Virtual Environment **https://ieeexplore.ieee.org/document/9070826**