

End-To-End Learning for Self Driving Cars

By: Bhuva, Vijay (86621)

Shah, Mihir (87568)

Why End-to-End Learning ?

- End-to-End Learning approach for self driving cars is one of the most successful amongst all
- There were other approaches, e.g. reinforcement learning
- But End-to-End learning, takes in several live data inputs of scenarios and sensors of car and predicts steering angle
- While reinforcement learning method, training is done while it explores new areas and later exploits all the experience it has gained during training
- Deep Janus a sample end to end learning algorithm available on GitHub helped a lot to design training of AI model for collection of data using Beamng.

Approach: Deep Janus and End to End CNN

Our approach follows following algorithm:

- Takes input as images from camera
- Uses deep neural networks as preprocessing images
- Abstracting only the areas of interest
- Predicting steering angle using CNN

Collection of data

- 3 camera arrangements[Left, Centre, Right]
- Images are stored into a log file along with indexes
- A csv file is created for image location, steering value, throttle value, clutch value
- Ai Span mode with 4 varying speed in Beamng
- Speed limit at 30 km/h, 50 km/h
- Expert drive
- Ai span mode available using beamngpy

Requirement.txt

ite Code Refactor Run Tools VCS Window DB Navigator MaxCompute Help Beamng_AI [C:\Users\HP\PycharmProjects\Beamng_AI] - _\batch_generator.py - PyCharm

erator.py

Settings

Project: Beamng_AI > Project Interpreter For current project Reset

Project Interpreter: Python 3.7 (venv) C:\Users\HP\PycharmProjects\deepjanus\venv\Scripts\python.exe

Package	Version	Latest version
Click	7.0	▲ 7.1.1
Flask	1.1.1	1.1.1
Jinja2	2.10.3	▲ 2.11.1
Keras-Applications	1.0.8	1.0.8
Keras-Preprocessing	1.1.0	1.1.0
Markdown	3.1.1	▲ 3.2.1
MarkupSafe	1.1.1	1.1.1
Pillow	7.0.0	7.0.0
PyOpenGL	3.1.5	3.1.5
Shapely	1.6.4.post2	▲ 1.7.0
Werkzeug	0.16.0	▲ 1.0.0
absl-py	0.9.0	0.9.0
astor	0.8.1	0.8.1
beamngpy	1.15	1.15
cycler	0.10.0	0.10.0
descartes	1.1.0	1.1.0
dill	0.3.1.1	0.3.1.1
drivebuild-client	0.40	0.40
gast	0.2.2	▲ 0.3.3
google-pasta	0.1.8	▲ 0.2.0
grpcio	1.26.0	▲ 1.27.2

Project: Beamng_AI > Project Interpreter For current project Reset

Project Interpreter: Python 3.7 (venv) C:\Users\HP\PycharmProjects\deepjanus\venv\Scripts\python.exe

Package	Version	Latest version
grpcio	1.26.0	▲ 1.27.2
h5py	2.10.0	2.10.0
itsdangerous	1.1.0	1.1.0
kiwisolver	1.1.0	1.1.0
matplotlib	3.1.2	▲ 3.2.0
msgpack	0.6.2	▲ 1.0.0
np	1.0.2	1.0.2
numpy	1.18.1	1.18.1
opt-einsum	3.1.0	▲ 3.2.0
pandas	1.0.0	▲ 1.0.2
pip	19.3.1	▲ 20.0.2
protobuf	3.11.2	▲ 3.11.3
pyparsing	2.4.6	2.4.6
pyshp	2.1.0	2.1.0
python-dateutil	2.8.1	2.8.1
pytz	2019.3	2019.3
scipy	1.4.1	1.4.1
setuptools	40.8.0	▲ 46.0.0
six	1.13.0	▲ 1.14.0
tensorboard	1.15.0	▲ 2.1.1

© 2014 Wolfram Research, Inc. All rights reserved.



A screenshot from the video game Grand Theft Auto: San Andreas. The scene is a street-level view looking down a road. On the left, there's a palm tree and a building with a red sign. In the center, a large billboard structure stands on the road. To the right, a large bridge or overpass curves over the road. The sky is blue with some clouds. The overall style is that of a 2000s-era open-world action game.

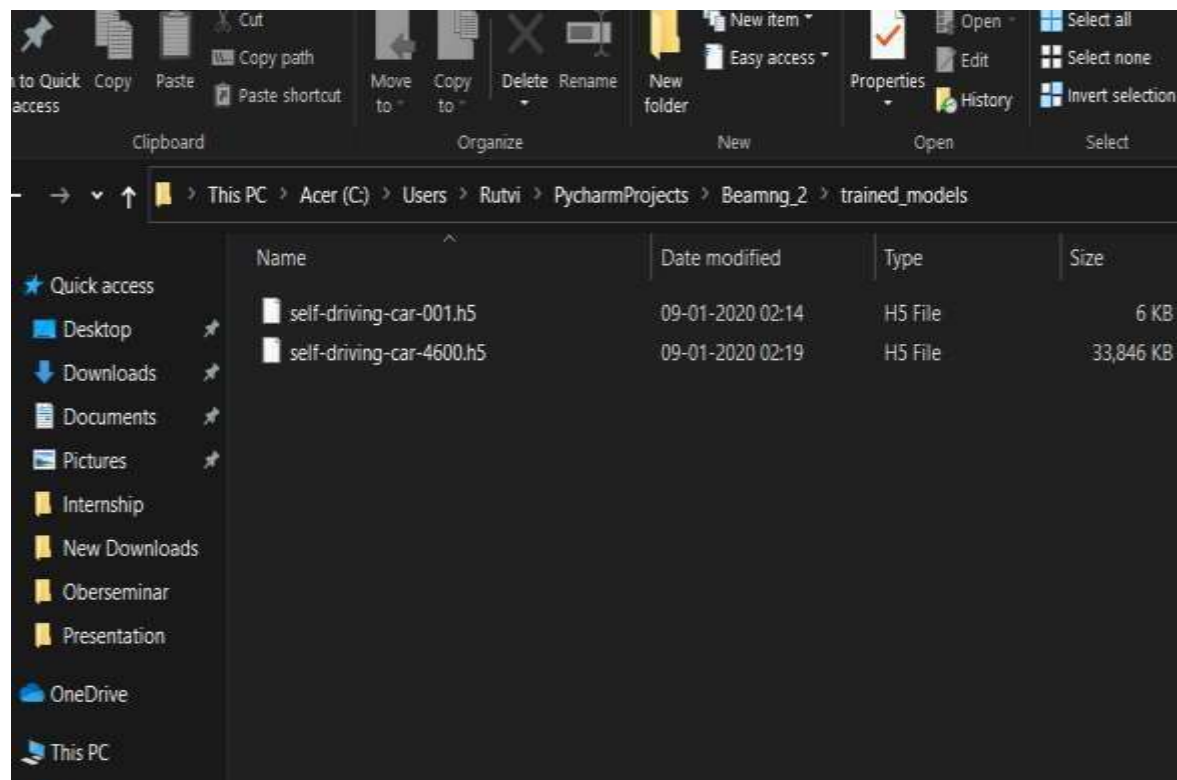


File
20 x 160

99_cam_center.jpg

[illegible]

Results of Training of Algorithm



Skipping registering GPU devices...

2020-01-09 05:45:14.196829: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device Interconnect StreamExecutor with strength 1 edge matrix:

2020-01-09 05:45:14.197102: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165] @

2020-01-09 05:45:14.197306: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] @

WARNING:tensorflow:From C:\Users\Rutvi\PycharmProjects\Beamng_2\venv\lib\site-packages\keras\backend\tensorflow_backend.py:172: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From C:\Users\Rutvi\PycharmProjects\Beamng_2\venv\lib\site-packages\keras\backend\tensorflow_backend.py:181: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From C:\Users\Rutvi\PycharmProjects\Beamng_2\venv\lib\site-packages\keras\backend\tensorflow_backend.py:188: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

1/50 [.....] - ETA: 3:50 - loss: 2021.1343

2/50 [>.....] - ETA: 2:48 - loss: 1902.3679

3/50 [>.....] - ETA: 2:25 - loss: 1888.8605

4/50 [=>.....] - ETA: 2:13 - loss: 1768.8985

5/50 [=>.....] - ETA: 2:05 - loss: 1698.3371

6/50 [=>.....] - ETA: 2:00 - loss: 1676.3506

7/50 [==>.....] - ETA: 1:55 - loss: 1729.9979


```

ego_vehicle.attach_sensor('cam_center', cam_center)
ego_vehicle.attach_sensor('cam_left', cam_left)
ego_vehicle.attach_sensor('cam_right', cam_right)
ego_vehicle.attach_sensor("electrics", Electrics())
#electrics_data = Electrics.encode_vehicle_request(Electrics)
#print(electrics_data)
scenario.add_vehicle(ego_vehicle, pos=(-717.121, 101, 118.675), rot=(0, 0, 4)
scenario.make(bng)
bng.open(launch = True)

def save_image(data, ind, cam_name):
    img = data[cam_name]['colour'].convert('RGB')

    file_name = str(ind) + "_" + cam_name + ".jpg"
    filepath = os.path.join('C:/Users/Rutvi/Desktop/Data_Log/Present_Log',
                            file_name)
    img.save(filepath)
    return file_name

def save(data, ind):
    cam_left_file_name = save_image(data, ind, 'cam_left')
    cam_right_file_name = save_image(data, ind, 'cam_right')
    cam_center_file_name = save_image(data, ind, 'cam_center')
    steering_in_value = data['electrics']['values']['steering_input']
    steering_value = data['electrics']['values']['steering']
    throttle_in_value = data['electrics']['values']['throttle_input']
    throttle_value = data['electrics']['values']['throttle']
    clutch_value = data['electrics']['values']['clutch']
    clutch_in_value = data['electrics']['values']['clutch_input']
    wheel_speed_value = data['electrics']['values']['wheelspeed']
    rpmspin_value = data['electrics']['values']['rpmspin']
    #add here

```

Data Collection

```

exit()

print("Train dataset: " + str(len(X_train)) + " elements")
print("Test dataset: " + str(len(X_valid)) + " elements")
return X_train, X_valid, y_train, y_valid

def build_model(args):
    """
    Modified NVIDIA model
    """
    model = Sequential()
    model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, (5, 5), activation='elu', strides=(2, 2)))
    model.add(Conv2D(36, (5, 5), activation='elu', strides=(2, 2)))
    model.add(Conv2D(48, (5, 5), activation='elu', strides=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='elu'))
    model.add(Conv2D(64, (3, 3), activation='elu'))
    model.add(Dropout(args.keep_prob))
    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))
    model.add(Dense(1))
    model.summary()

    return model

def train_model(model, args, X_train, X_valid, y_train, y_valid):
    """
    Train the model
    """

```

Trained Model

Test Generator

- For test generator, I tried to implement, miscellaneous tracks
- Tracks includes combination of single, double lane tracks
- Also, at some location obstacles such as cone, cylinder were installed
- The objective of such test generator was to test the out of bound episodes of our trained AI
- Also to test the AI under stressful situation

Thank You