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

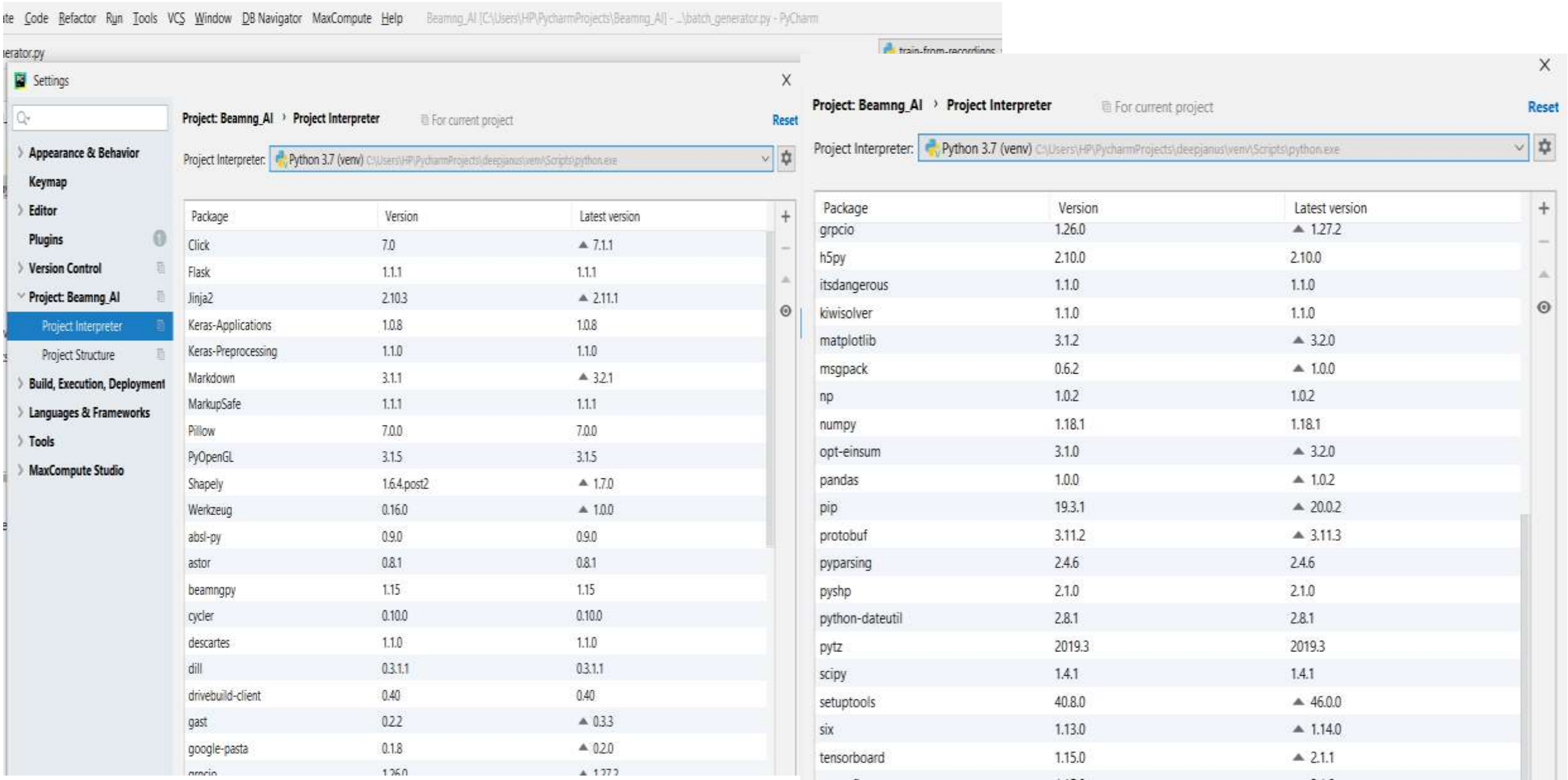
Our approach follows following algorithm:

- 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

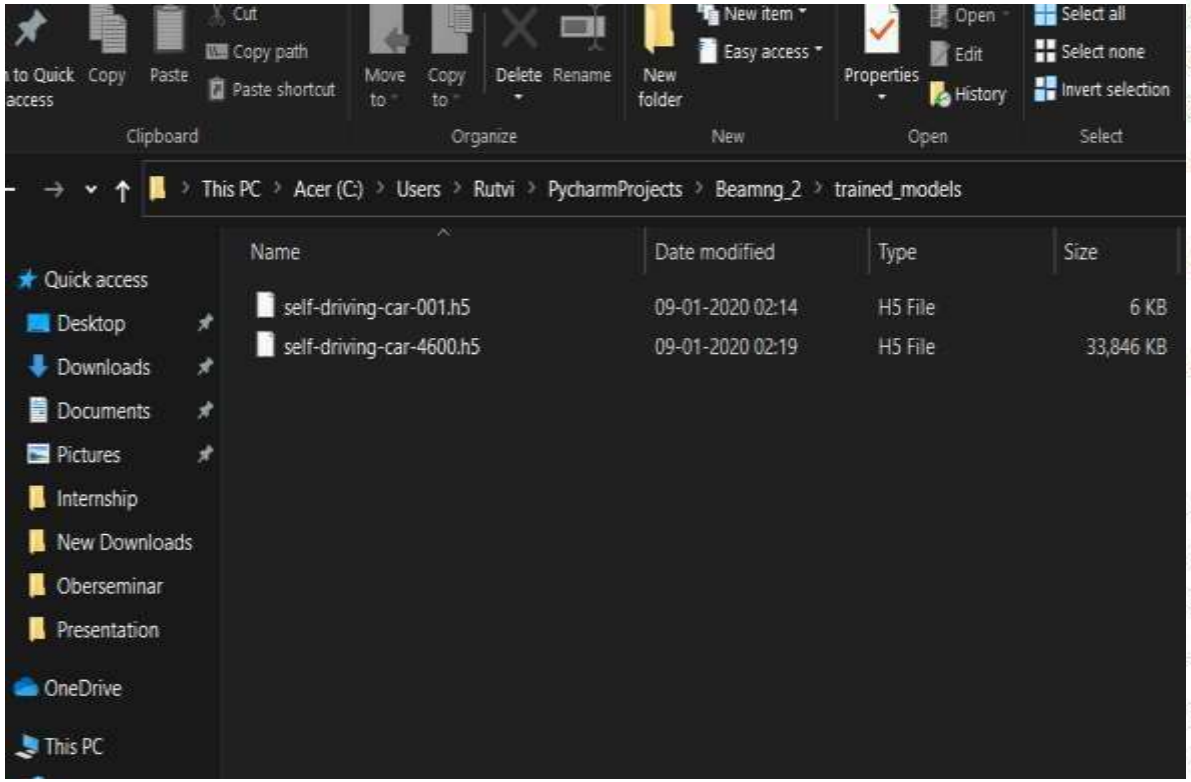
- 3 camera arrangements[Left, Centre, Right]
- Images are stored into a log file along with indexes
- A csv file is created for image location, steering value, throttle value, clutch value
- Ai Span mode with 4 varying speed in Beamng
- Speed limit at 30 km/h, 50 km/h
- Expert drive
- AI span mode available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

Requirement.txt



Results of Collection of Data

Results of Training of Algorithm



```
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, 45
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