Objects based imitation learning home assignment

This home assignment aims to gauge the degree to which the candidate can build a deep learning objects-list based model for autonomous driving, a data pipeline, and a train script using the PyTorch library and some training data we provide.

There is not enough training data to produce a high performing model but the architecture and training algorithm should be such that with more data, the model would have a chance to achieve decent performance. We provide data on detected objects, detected lanes, imu data, and waypoints. The model should take information from the detected objects, detected lanes and imu data as input and output 4 waypoints per forward pass similar to the ones in the training data.

The backbone of the model can be of any kind. The head of the model should output a tensor of shape (B, 4, 2) per forward pass (representing 4 2d waypoints and where B is the batch dimension) and should be based on an RNN architecture (GRU, LSTM etc.), i.e. each 2d waypoint should be successively returned by the RNN until all 4 waypoints have been returned. You are free to use built-in modules from PyTorch but you should be able to answer questions about their inner workings.

Data has been provided for 2 unique 1 minute long scenarios. Each scenario has waypoints and 3 object files associated with it:
imu_data.json contains IMU information about the ego vehicle for every frame.
cametra_interface_lanes_output.csv contains information about detected lanes for each frame
cametra_interface_output.csv contains information about all detected objects in each frame.

For each unique timestamp there will be multiple rows corresponding to the lanes and objects detected in that frame. Note that the column “name” in c*ametra_interface_output.csv* and “frame_id” in *cametra_interface_lanes_output.csv* contain the same timestamps. The keys of *imu_data.json* also contain the same timestamps. Lastly keep in mind that if no lanes or objects are detected for a particular timestamp there will be no data provided in the csv file. As such it is advisable to use the timestamps provided in *imu_data.json* as reference as those offer a complete set of timestamps for a given scenario.

Feel free to use any features for the model and make sure you can justify their use. A simple model could for example use the lat_dist, long_dist, abs_vel_x, abs_vel_z features from *cametra_interface_output.csv*, boundary_name, polynomial [0], polynomial [1], polynomial [2] from *cametra_interface_lanes_output.csv* and vf feature from *imu_data.json* for every timestamp. Make sure that your model can handle variable input sizes as the number of objects detected in each frame will vary.

Each scenario contains 120 timestamps. Meaning that there are at most 120 unique timestamps in *cametra_interface_lanes_output.csv* and in *cametra_interface_output.csv.* The *waypoints.npy* array provided is of shape (120, 4, 2). Thus for each timestamp there is an associated (4, 2) waypoints array. The 4 waypoints are spaced 0.5s apart and are described by (lateral_distance, longitudinal_distance) from the perspective of the current ego position and orientation, in meters.

Images have been provided to help you visualize the scenarios but they are not necessary for completion of the assignment.

The data directory structure is shown below:

```
data/
├── images/
│   └── sequence_name/
│       ├── img_1.png
│       ├── img_2.png
│       ├── ...
│       ├── ...
│       └── img_120.png
├── waypoints/
│   └── sequence_name/
│       └── waypoints.npy
└── objects/
    └── sequence_name/
        ├── cametra_interface_lanes_output.csv
        ├── cametra_interface_output.csv
        └── imu_data.json
```

Please write clean code with a reasonable file structure and include a readme file as well as a requirements file to facilitate running your solution.