Aalto University
School of Electrical
Engineering

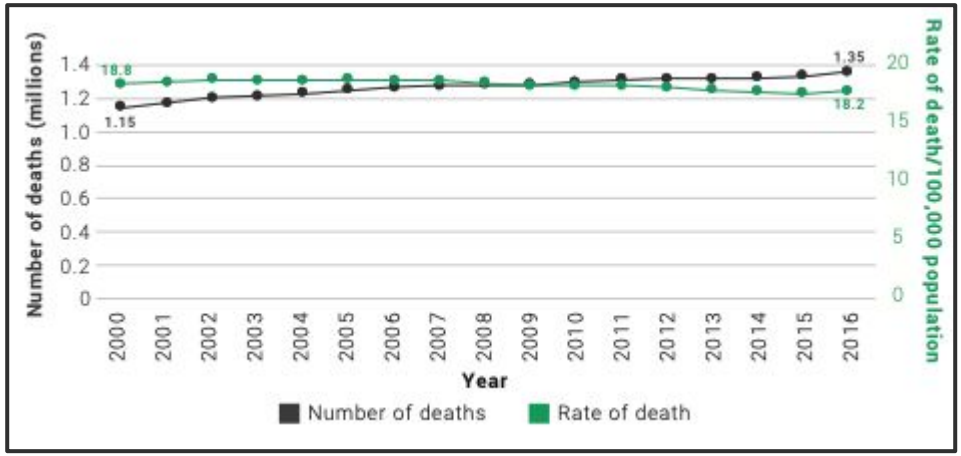
VOLVO
Volvo Group Trucks

Highway autopilot using DRL and GNNs

Harikrishnan Devaraj

Supervisor : Prof. Kyrki Ville
Advisors : Börve Erik, Gökhan Alcan, Pathare Deepthi

Motivation

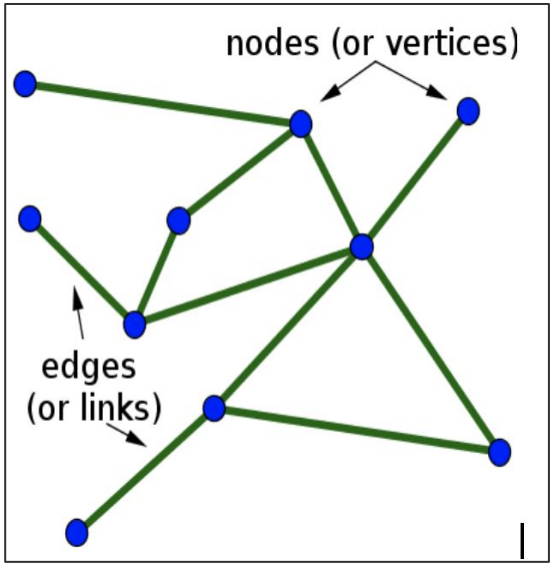


- **1.35 million people** die each year as a result of road traffic crashes - WHO report (2018)
- Human error attributed to the root cause .
- One possible way to tackle this would be to have a semi or fully autonomous system which augments the driver to make better decisions.

Results from Literature review

- Most of the current papers published in this domain focuses on **discrete action space**.
- In the real world, vehicles on the road can change rapidly from the point of view of the truck/ego vehicle.
- For NN, need to know the maximum number of vehicles and in which order to sense.
- CNNs using occupancy grid limited to initial grid size.
- RNNs usefully to cover temporal context, but not invariant wrt to input order for a fixed time step.
- **Why use Graph neural networks over other NNs -**

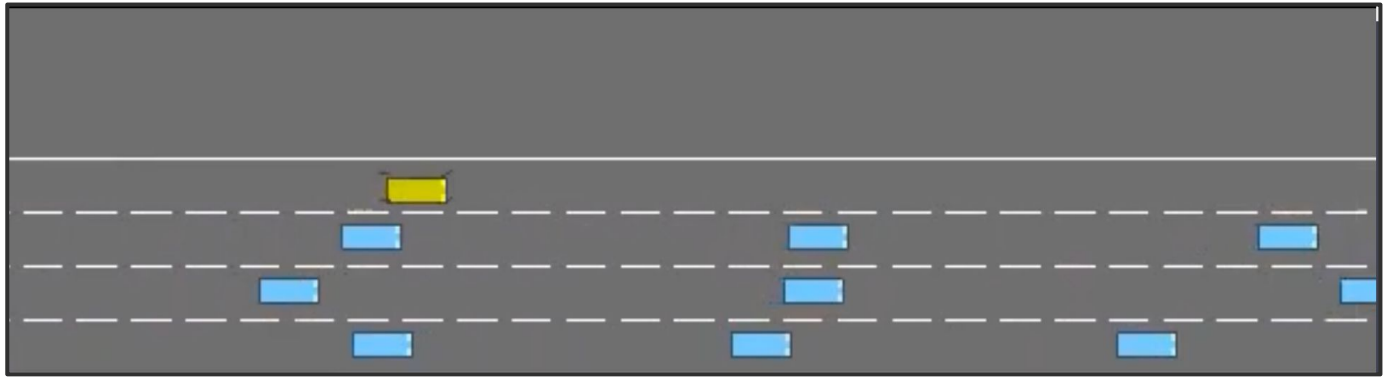
  GNNs are invariant to number and order of vehicles, introduce relational bias explicitly to the learning problem.



Goals of this thesis

- Investigate the feasibility of utilizing DRL together with a GNN to accomplish a safe driving scenario in a continuous action space.
- **Research questions(RQ)** :
  - Investigate if there is a change in performance on the model depending on the input graph structure.
  - Investigate possibility of knowing the decision making process of the policy using GNN.

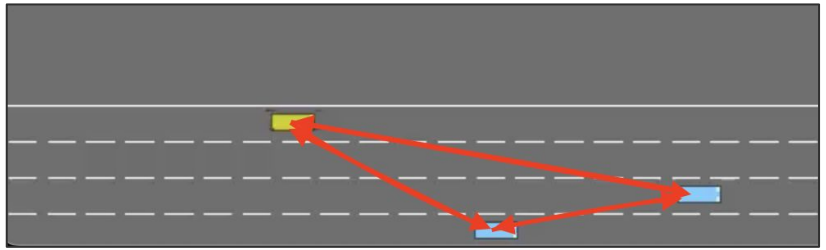
Environment - HighwayEnv

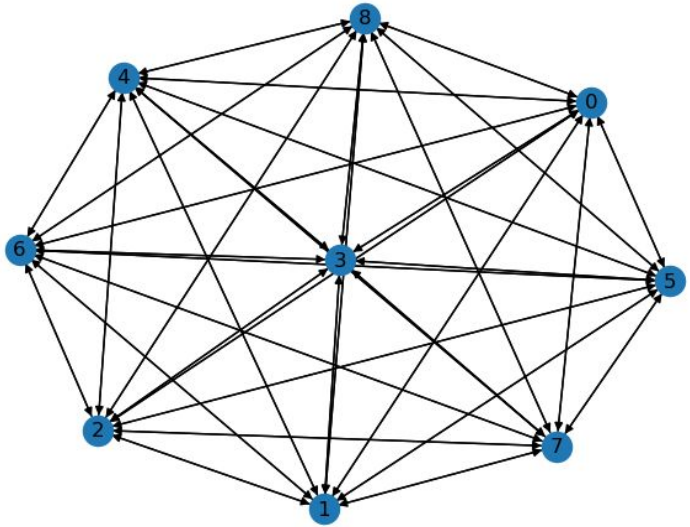


- Multi-laned highway environment - “highway-v0”
- Low level Control - acceleration and steering angle.
- [-3.5,3.5] m/s^2 , [-10, 10] radians - continuous range

Graph creation

- Uses *KinematicObservaton* array output.
- Constructs graph within a perception distance of 200m.
- Individual vehicles are nodes.
- Node 0 is the ego-vehicle.
- Edge attributes contain relative information.
- Dynamically updates if a new vehicle comes within the perception distance.





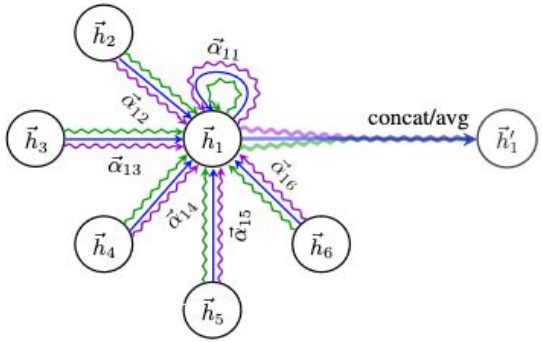
Node and Edge attributes - 11 and 5

Node Attribute	Definition
$presence$	Disambiguate agents at 0 offset from non-existent agents.
x	World offset of ego vehicle or offset to ego vehicle on the x axis.
y	World offset of ego vehicle or offset to ego vehicle on the y axis.
vx	The velocity component in the x-direction.
vy	The velocity component in the y-direction.
$heading$	Heading of the vehicle in radians.
cos_h	Trigonometric heading of vehicle.
sin_h	Trigonometric heading of vehicle
cos_d	Trigonometric direction to vehicle's destination.
sin_d	Trigonometric direction to vehicle's destination.
$long_{off}$	Longitudinal offset to closest lane.
lat_{off}	Lateral offset to closest lane.
ang_{off}	Angular offset to closest lane.

Edge Attribute	Definition
$delta\_x$	The change in the x-coordinate between two vehicles.
$delta\_y$	The change in the y-coordinate between two vehicles.
$delta\_vx$	The change in velocity component in the x-direction between two vehicles.
$delta\_vy$	The change in velocity component in the y-direction between two vehicles.
$delta\_theta$	The change in heading angle between two vehicles.

Graph attention network structure - GNN part

- GNN with Self-attention.
- Two GATv2 layers.
- One dense layer.
- Tanh function.
- Only ego-vehicle embeddings given to dense layer.
- Using PyG library




```
) SoftGATv2Network(
  (conv1): GATv2Conv(11, 55, heads=5)
  (conv2): GATv2Conv(-1, 110, heads=1)
  (dense): Sequential(
    (0): Linear(in_features=110, out_features=12, bias=True)
    (1): Dropout(p=0.8, inplace=False)
    (2): Linear(in_features=12, out_features=275, bias=True)
  )
)
```

Soft actor -critic algorithm - DRL part

- Actor -critic based RL algorithm.
- One Actor and Two critic networks.
- Continuous action spaces.
- Code based on Clean RL implementation.

Actor Network architecture

- GNN layer
- Linear layer (275x256)
- Linear layer(256x256)
- Final layer(256x2)
- Adam optimizer

```
Actor_GNN(
(gat_gnn_actor): SoftGATv2Network(
  (conv1): GATv2Conv(11, 55, heads=5)
  (conv2): GATv2Conv(-1, 110, heads=1)
  (dense): Sequential(
    (0): Linear(in_features=110, out_features=12, bias=True)
    (1): Dropout(p=0.8, inplace=False)
    (2): Linear(in_features=12, out_features=275, bias=True)
  )
)
(fc1): Linear(in_features=275, out_features=256, bias=True)
(fc2): Linear(in_features=256, out_features=256, bias=True)
(fc_mean): Linear(in_features=256, out_features=2, bias=True)
(fc_logstd): Linear(in_features=256, out_features=2, bias=True)
```

Critic Network architecture

- GNN layer
- Linear layer(277x256)
- Linear layer(256x256)
- Linear layer(256x1)
- Adam optimizer
- Two critics

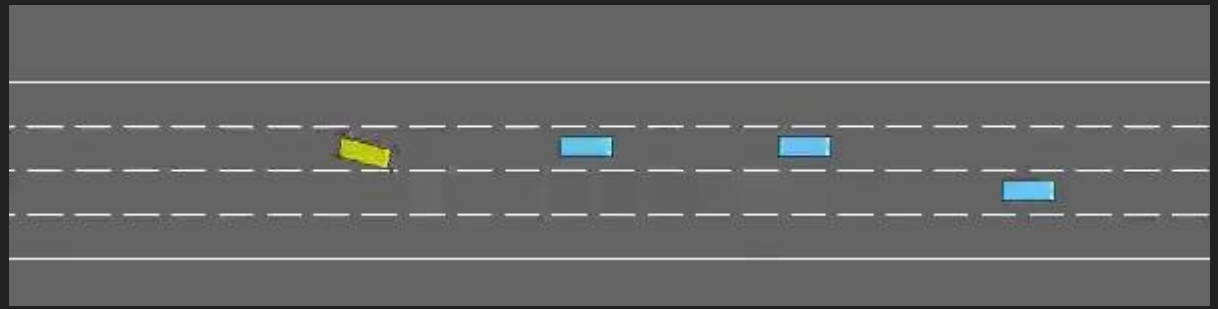
```
) SoftQNetwork_GNN(
  (gat_gnn_critic): SoftGATv2Network(
    (conv1): GATv2Conv(11, 110, heads=10)
    (conv2): GATv2Conv(-1, 220, heads=1)
    (dense): Sequential(
      (0): Linear(in_features=220, out_features=1, bias=True)
      (1): Dropout(p=0.3, inplace=False)
      (2): Linear(in_features=1, out_features=275, bias=True)
    )
  )
  (fc1): Linear(in_features=277, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=1, bias=True)
)
```

Tasks

- Train RL agent in the continuous action space using Highway Env(lane change).
- Generate the graph structure with respect to the ego vehicle .
- Implement the Actor and Critic networks(NNs) in SAC as GNNs.

Current results with Soft actor critic(SAC) - Time steps -200,000

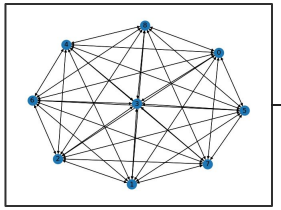
acceleration_range" : [-3.5,3.5],
"steering_range" : [-0.7853981633974483, 0.7853981633974483],



Timesteps = 1 million,

'high_speed_reward': 0.5,
'on_road_reward' : 0.1,
'collision_reward': -1, "lane_change_reward": 0.1,   'reward_speed_range': [20, 25],

Soft GATv2 Actor network(Graph attention actor network)



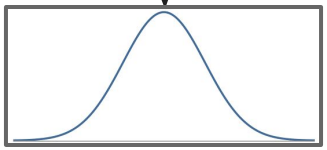
Bidirectional input graph

SoftGATv2(GNN)

New ego embeddings

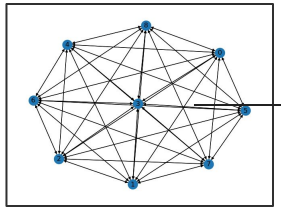
Batch size 256

Actor NN layers



Action distribution- [acceleration, steering angle]

SoftGATv2 Critic networks(n_critics = 2 fo SAC)



Bidirectional input graph

SoftGATv2(GNN)

New ego embeddings

Actions
(Actor network)

||

Concatenate

Critic NN layers

State-action value estimate

Reward function and weight values

Main objectives -

- Try to travel fast.
- Avoid collisions.
- Try to stay on the right lane.
- Be on the road.

$$R(s,a) = h\left(\frac{v - v_{min}}{v_{max} - v_{min}}\right) + b(\text{collision}) + c(r_{rightlane}) + r_{onroad}$$

$$\text{collision} = \begin{cases} 1 & \text{if there is a collision} \\ 0 & \text{if there is no collision} \end{cases}$$

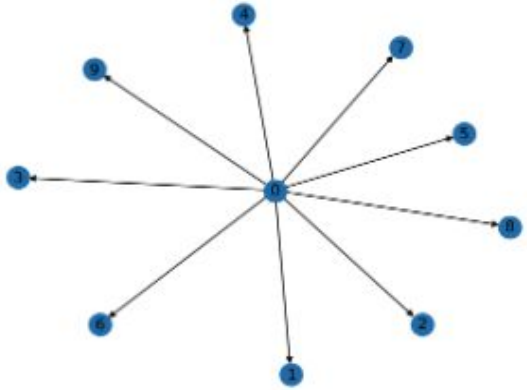
$$r_{rightlane} = \frac{\text{Current Lane Index}}{\max(\text{Total Number of Neighboring Lanes} - 1, 1)}$$

$$r_{\text{onroad}} = \begin{cases} 1 & \text{if the vehicle is on the road} \\ 0 & \text{if the vehicle is off the road} \end{cases}$$

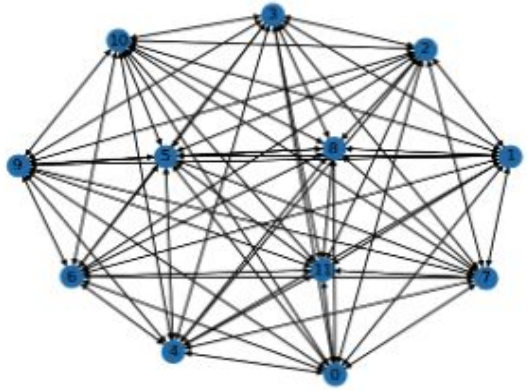
Reward weight	value
high speed reward weight(h)	0.5
collision penalty weight(b)	-1
right lane reward weight(c)	0.2

Change in performance with two input graphs - RQ1

- Investigate if there is a change in performance for two different input graphs during training phase and testing phase.
- For random seed value - 42 , trained for 1e6 timesteps.

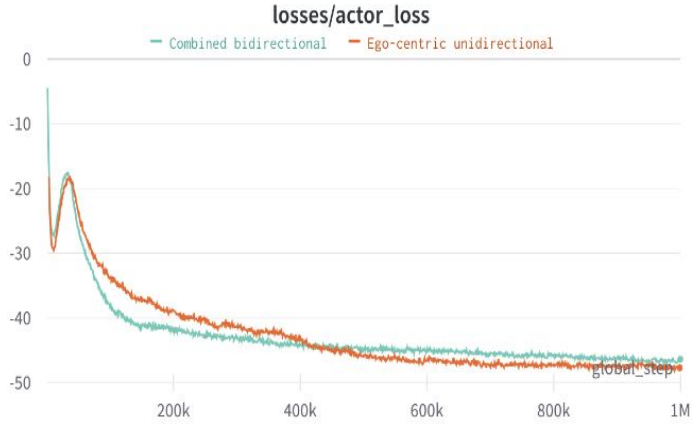


(a) Ego-centric unidirectional.

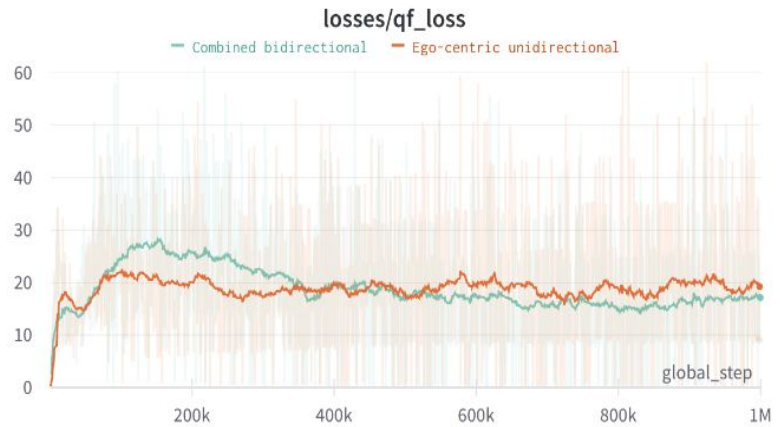


(b) Combined vehicle bidirectional.

Training phase - loss plots

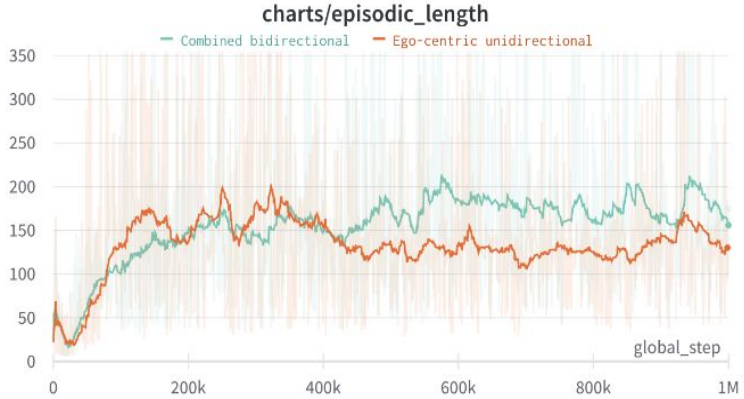


(a) SoftGATv2 Actor network loss vs Global step.

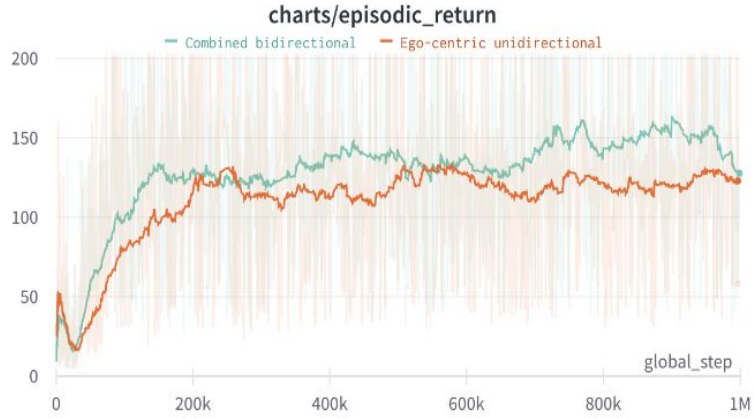


(b) SoftGATv2 Critic network loss vs Global step.

Training phase

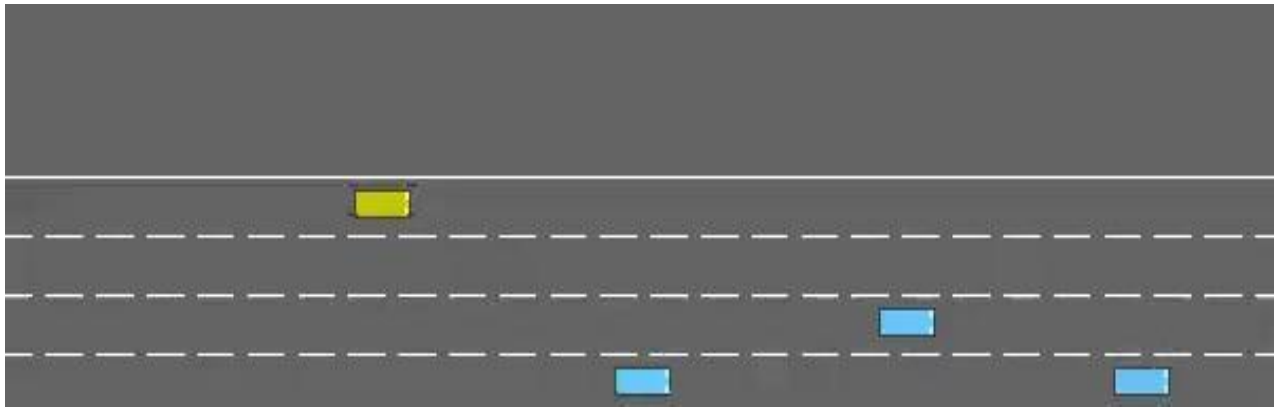


(a) Episodic length vs Global step.



(b) Episodic reward return vs Global step.

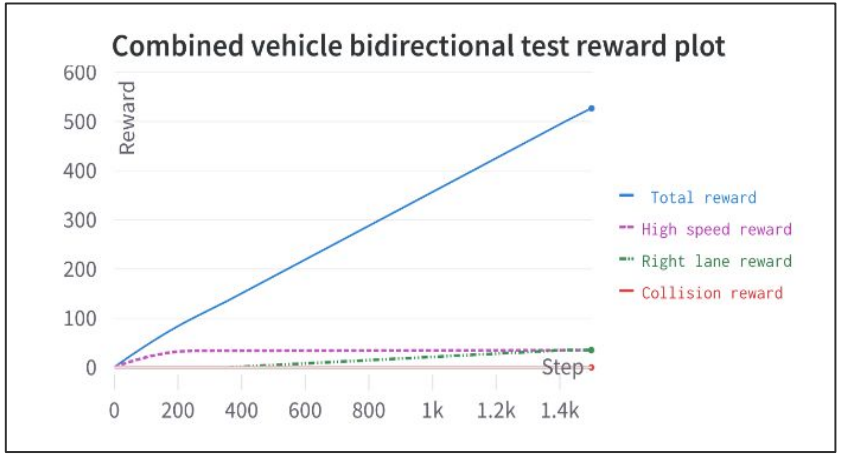
Ego-centric unidirectional

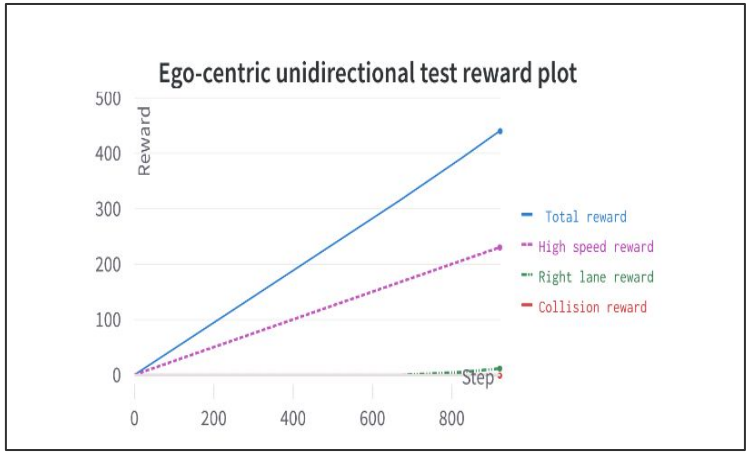


Combined Vehicle bidirectional

Testing Phase

- There is a change in performance, but which input graph is better can not be concluded due to sparsity of the result with a single initial condition.

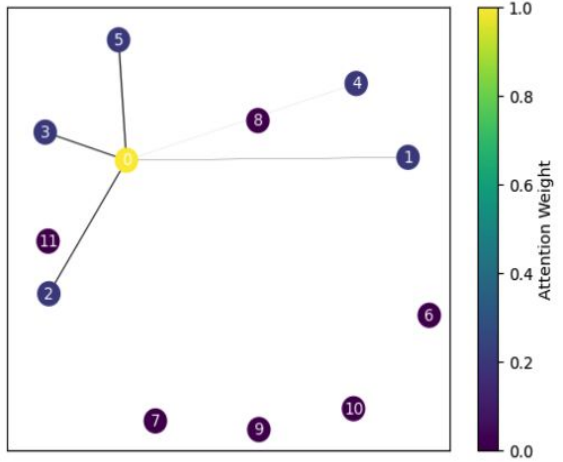




Weighted test reward	Ego-centric Unidirectional	Combined Vehicle Bidirectional
Total reward	439.965	526.76386
High speed reward	230.2405	34.96523
Right lane reward	11.6	35.53333
Collision penalty	-1	0

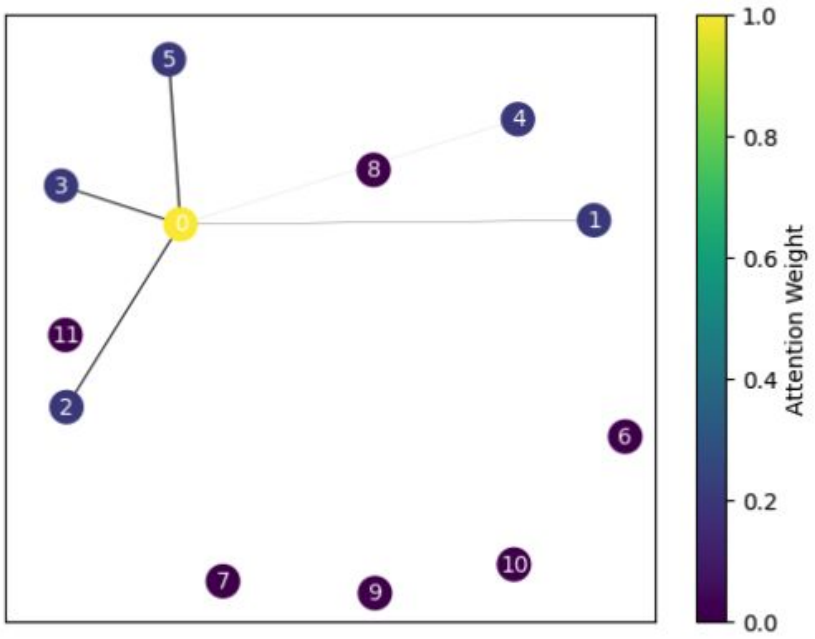
Policy network attention weights - RQ2

- Attention weights visualized for an initial instant of the test simulation for GATv2 layers.
- The node number denotes vehicles in the order of the proximity to node 0.
- Node number is same as the row number in the *KinematicObservation* array output.
- More the thickness of the edge and higher the color in the scale, more attention is given.
- Ego-vehicle gives importance to vehicles near it for this initial instant.
- Can be used for any instant in the simulation.

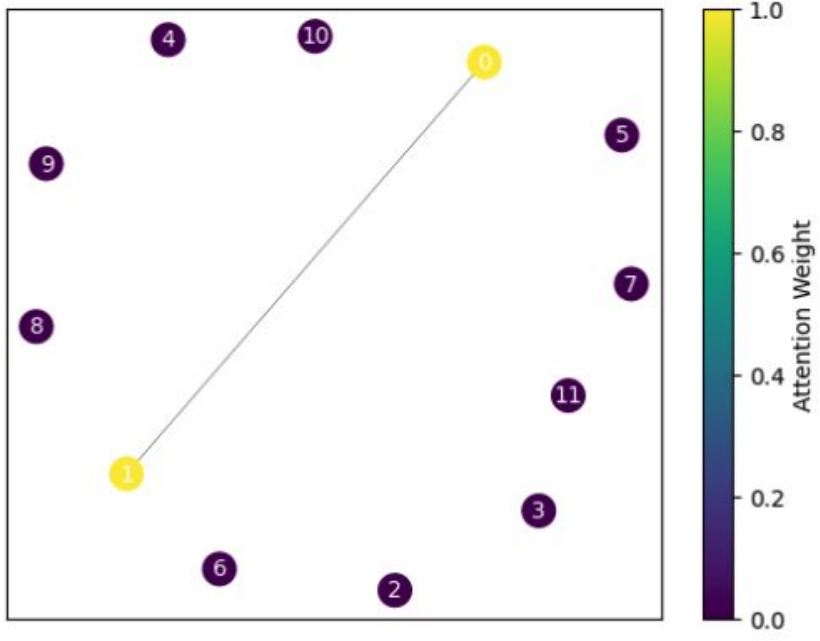


(a) Attention weight connections in SoftGATv2 layer 1 for ego-vehicle node 0.

Policy network attention weights for the two layers



(a) Attention weight connections in SoftGATv2 layer 1 for ego-vehicle node 0.



(b) Attention weight connections in SoftGATv2 layer 2 for ego-vehicle node 0.

Conclusions

- There is a change in performance of the model depending on the input graph, but which input graph is optimal can not be concluded due to sparsity of result with a single initial condition.

- Policy network attention weights of GATv2 can be visualized to know some part of the decision making process of the model.

Limitations and Future work

- **RQ1 only have results for a single configuration of the highway environment**.

  So, can not make conclusion for which input graph is better.

  Perform more experiments with different environment configurations to know which input graph structure is optimal. Also include other input graph structures.

- **Long training time for the model - around 5 days**.

  Reduce training time using multiple GPUs or other parallelization methods.

- Use a different scenario like intersection to see the usefulness of GNN in a high vehicle density environment.
- Use different algorithm for the DRL and GNN components and find the best model for a given environment scenario.

THANK YOU

```
action": {
            "type": "ContinuousAction",
            "acceleration_range" : [-3.5,3.5],
            "steering_range" : [-0.7853981633974483, 0.7853981633974483],
            'lateral' : True,
            'longitudinal': True,
            "speed_range" : [10,25],
"normalize_reward" : True,
“see_behind “ :True
Rewards
            'high_speed_reward': 0.5,
            'on_road_reward' : 0.1,
            'collision_reward': -1,
            #'right_lane_reward' : 0.1,
            #"high_speed_reward": 0.5,  # The reward received when driving at full speed, linearly mapped to zero for
                                       # lower speeds according to config["reward_speed_range"].
            "lane_change_reward": 0.1,
            'reward_speed_range': [20, 25],
```