Capstone Project

Shubham S PES1UG20CS420
Shuchith B U PES1UG20CS421
Siddarth M P PES1UG20CS423
Amogh N Rao PES1UG20CS625

Project Guide: BhaskarJyoti Das
Project ID :PW_23_BJD07

PES UNIVERSITY

EMULATING EMERGENCY VEHICLE NAVIGATION IN MIXED TRAFFIC ENVIRONMENT USING GRAPH PREDICTION AND SIMULATION

PES UNIVERSITY

PROBLEM STATEMENT

The integration of autonomous vehicles (AVs) into our traffic networks is happening quite quickly. Although completely autonomous driving is still a ways off, in the near future, traffic will be composed of both human-driven and autonomous vehicles

Learning representations of nodes in a graph is possible with the use of machine learning models called graph neural networks (GNNs). The efficacy of GNNs has been demonstrated in numerous traffic network-related tasks,

Lack of innovative method that uses GNNs for efficient EV navigation in mixed traffic environments. Our method tackles the problem of sensor failures in real life situations by utilizing GraphSAGE for link prediction

PES UNIVERSITY

SCOPE

Since autonomous vehicles (AVs) have been seamlessly incorporated into contemporary traffic environments, emergency vehicles (EVs) are required to navigate alongside their human-driven and autonomous counterparts

Although some bottleneck problems have been addressed by simulations based on reinforcement learning (RL), these approaches often fail to capture the intricate dynamics of real-world traffic conditions.

As an addition, a customized action policy function is presented, which uses GraphSAGE's projected link states to guide decision-making over lane changes

PES
UNIVERSITY

WHY GRAPHSAGE??

At times sensor failures can lead to data gaps, resulting in the absence of links between vehicles.

Link prediction can enhance this process by predicting the likelihood of connections or interactions between vehicles, which in turn can be used to optimize emergency response strategies.

Graphsage is responsible for generating node embeddings for vehicles. Node embeddings are vector representations of nodes in a graph that capture their unique characteristics and their relationships with neighboring nodes

PES UNIVERSITY

LINK PREDICTOR NN

This component is responsible for predicting the existence of links between vehicles. It takes the node embeddings generated by GraphSAGE as input and produces a link prediction score for each pair of nodes. The higher the link prediction score, the more likely it is that a link exists between the two corresponding nodes.

TEAM ROLES AND RESPONSIBLITIES

Having said about our Problem statement and focusing on solving it in phases , we divided our work equally

- One of the primary responsibility involved in generating SUMO scenarios and integrate through Traci library to produce graph dataset
- The other responsibility of looking through GrapSAGE and its advantages over other algorithms which makes it better to employ was looked in by other teammate

PES
UNIVERSITY

TEAM ROLES AND RESPONSIBLITIES

- The other aspect of Link Predictor Neural network and ways to implement it and how it can be significant in real world scenarios was carried out by other teammate

- And finally comparing our model with baseline models like Inbuilt IDM3 of SUMO, and without GraphSAGE models and drawing out the insights from it was carried by other teammate

SUMMARY OF PHASE - 1

Through extensive Literature survey being carried out, some of they key findings were :

- Graph data structure better represents the traffic environment, Hence making use of Graph Neural Networks Graphsage in particular for obtaining embeddings from neighbor nodes helps in link prediction.

- Some action policy need to be determined for efficient working of module, which can be RL or a custom action policy function.

PES UNIVERSITY

SUMMARY OF PHASE - 1

- Key Findings:

- Graph Neural Network models: GCN, GAT etc.
- Evaluration metrics: Waiting time, Collisions, Travel time, Average Speed etc
- Lack of handling sensor failure cases

- Baseline models to compare with:
- IDM3 algorithm only which does not use any custom action policy
- Custom action policy function that uses features obtained by GraphSAGE to predict links.

PES UNIVERSITY

SUMMARY OF PHASE - 1

Influence from literature review:

- As discussed, there has been some papers solving the problem of efficient traversal of various vehicles in mixed environments, not solely focused on one
- Application of GRL algorithm on fewer vehicles has been carried out which has been widely used in such scenarios for better representation of problem space

- **Designed Problem** :
- After going through literature survey, we narrowed down the problem to one single vehicle, particularly focusing on “Emergency vehicle”, upon which our research will suffice
- By comparing with various statistical measures, will measure efficient traversal of this vehicle in mixed environment by link prediction.

WORKFLOW

With the designed simulation environment ,Emergency vehicles are granted priority over other vehicles in the traffic network. This priority allows them to navigate through the traffic network faster, potentially reaching their destinations more quickly

Overall traffic efficiency, average speed, and navigation waiting time—are considered to assess the effectiveness of the proposed approach. Navigation waiting time measures the time an emergency vehicle spends waiting to reach its destination

The proposed approach is compared against two baseline methods. Baseline 1 relies solely on the IDM3 algorithm without utilizing any custom action policy. Baseline 2 incorporates a custom action policy function using simulation data obtained from the traCI library instead of GraphSAGE to predict links.

ARCHITECTURE

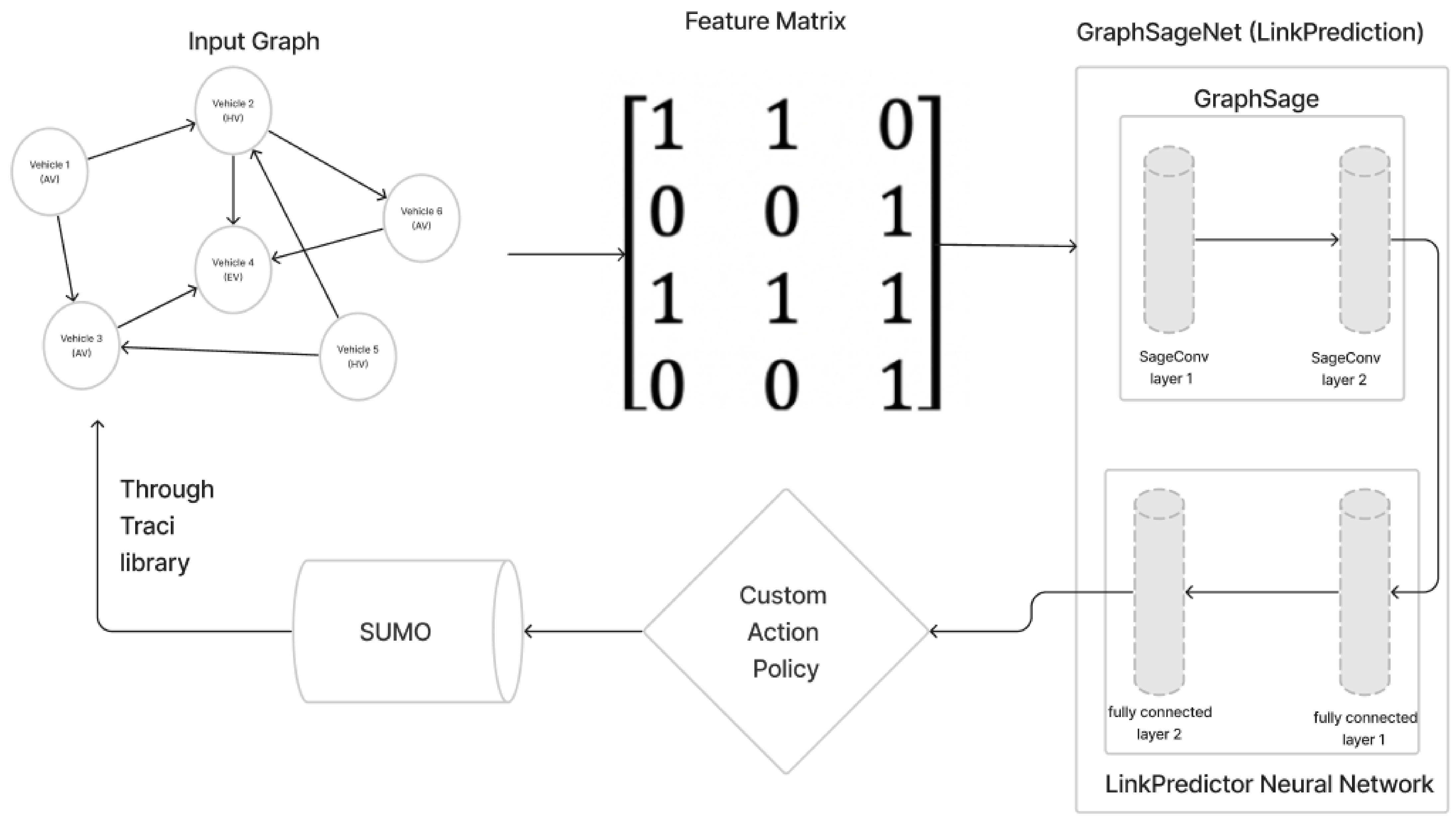
GraphSAGE model comprises two layers of SAGEConv, a variant of the GraphSage graph neural network (GNN). This architecture is designed to aggregate information from neighboring nodes, allowing the model to generate node representations that capture the interactions between vehicles in a graph

LinkPredictionNN is a model component with two connected layers, learning patterns from GraphSAGE embeddings to predict potential links between vehicles by generating hidden-dimensional representations and single output values that depicts potential links between vehicles.

The GraphSageNet model integrates the strengths of the GraphSAGE model and LinkPredictionNN, generating informative node embeddings for vehicle interactions and predicting missing links for improved performance.

PES UNIVERSITY

ARCHITECTURE



$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

METHODOLOGY

The GraphSageNet model is trained on a dataset obtained from the SUMO software, using the TraCI API. The dataset represents a traffic network extracted from OpenStreetMap data, encompassing X vehicles.

The model constructs a training graph based on the obtained data, and GraphSageNet is employed to predict missing links within the traffic graph. This involves leveraging the learned embeddings to identify potential connections between vehicles.

In the simulation process, GraphSageNet's predictions inform the decision-making process for autonomous and emergency vehicles. A Custom Action policy is applied to make informed decisions based on the predicted missing links, enhancing navigation and actions within the simulated traffic environment.

PES UNIVERSITY

IMPLEMENTATION

SIMULATION ENVIRONMENT

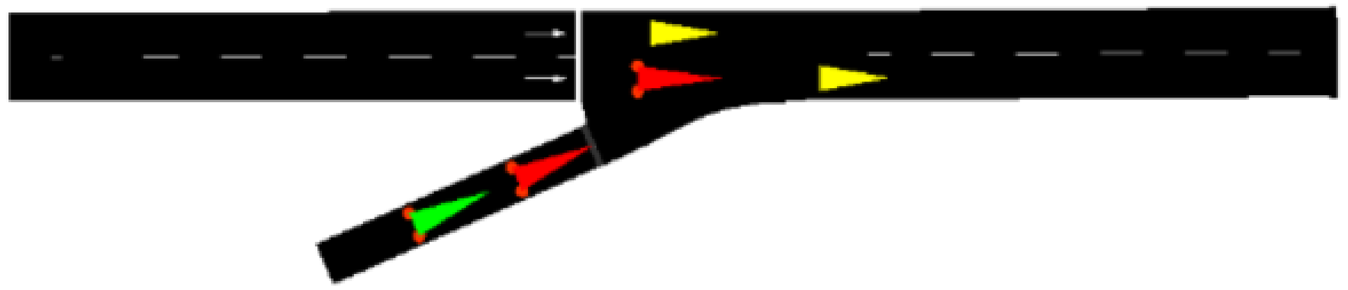


Fig. 1. Merge Network

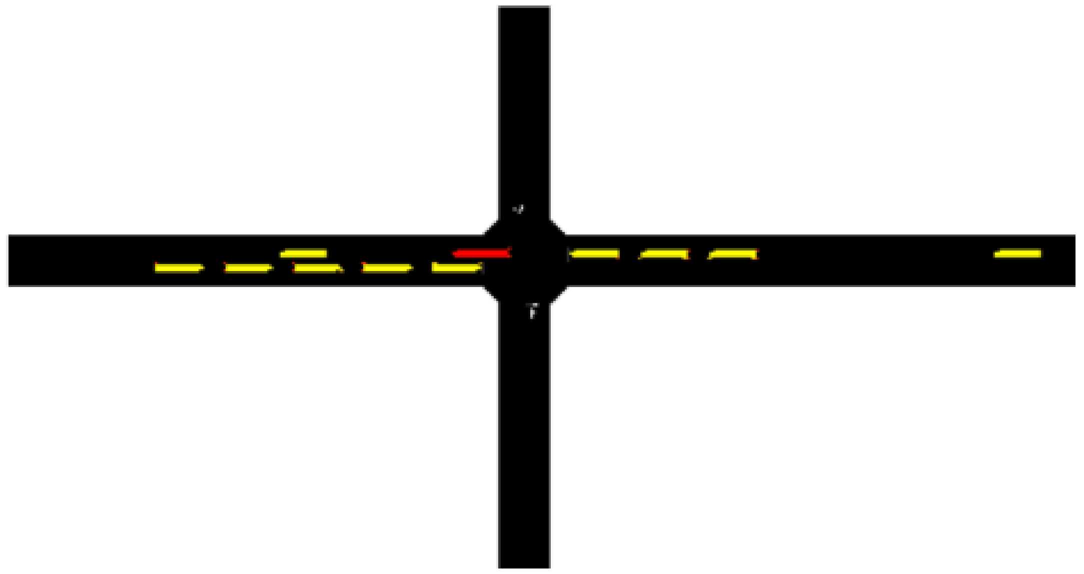


Fig. 2. Intersection Network

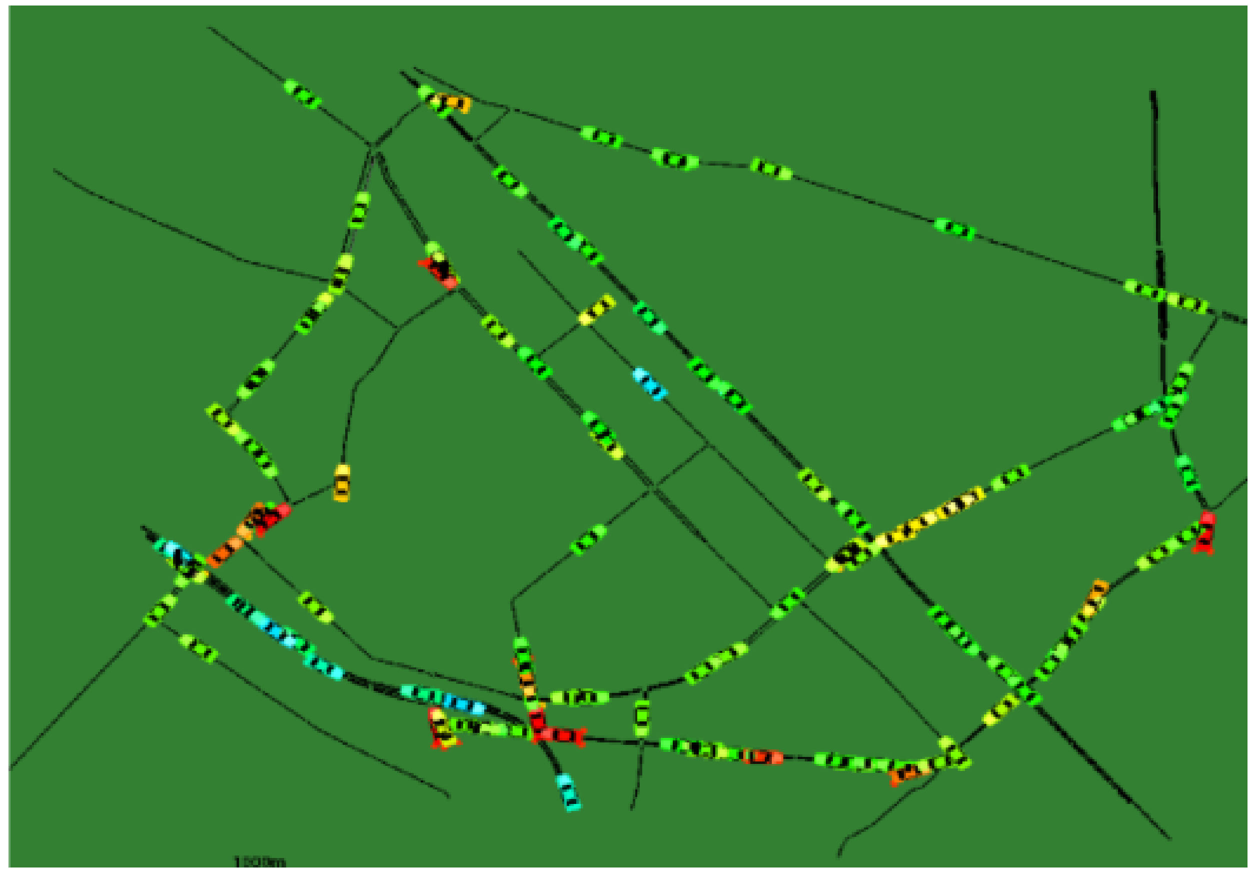
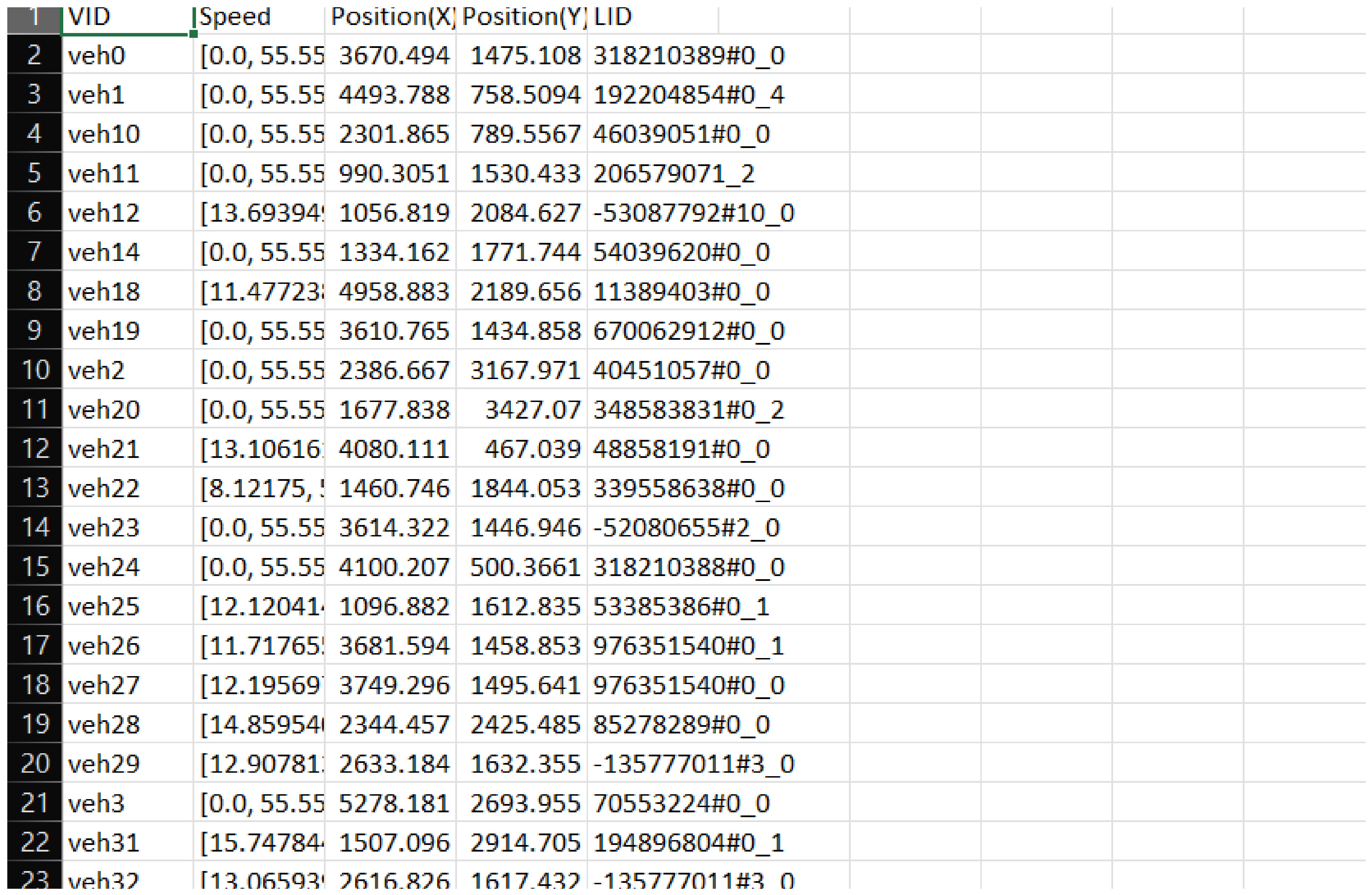


Fig. 3. City Network

DATASET

Node Features



	VID	Speed	Position(X)	Position(Y)	LID
1	VID	Speed	Position(X	Position(Y	LID
2	veh0	[0.0, 55.55	3670.494	1475.108	318210389#0_0
3	veh1	[0.0, 55.55	4493.788	758.5094	192204854#0_4
4	veh10	[0.0, 55.55	2301.865	789.5567	46039051#0_0
5	veh11	[0.0, 55.55	990.3051	1530.433	206579071_2
6	veh12	[13.69394	1056.819	2084.627	-53087792#10_0
7	veh14	[0.0, 55.55	1334.162	1771.744	54039620#0_0
8	veh18	[11.47723	4958.883	2189.656	11389403#0_0
9	veh19	[0.0, 55.55	3610.765	1434.858	670062912#0_0
10	veh2	[0.0, 55.55	2386.667	3167.971	40451057#0_0
11	veh20	[0.0, 55.55	1677.838	3427.07	348583831#0_2
12	veh21	[13.10616	4080.111	467.039	48858191#0_0
13	veh22	[8.12175,	1460.746	1844.053	339558638#0_0
14	veh23	[0.0, 55.55	3614.322	1446.946	-52080655#2_0
15	veh24	[0.0, 55.55	4100.207	500.3661	318210388#0_0
16	veh25	[12.12041	1096.882	1612.835	53385386#0_1
17	veh26	[11.71765	3681.594	1458.853	976351540#0_1
18	veh27	[12.19569	3749.296	1495.641	976351540#0_0
19	veh28	[14.85954	2344.457	2425.485	85278289#0_0
20	veh29	[12.90781	2633.184	1632.355	-135777011#3_0
21	veh3	[0.0, 55.55	5278.181	2693.955	70553224#0_0
22	veh31	[15.74784	1507.096	2914.705	194896804#0_1
23	veh32	[13.06593	2616.826	1617.432	-135777011#3_0

DYNAMIC DATASET

The dataset originates from real-life situations in OpenStreetMap (OSM), where specific areas of interest, such as junctions, are identified and focused on.

This scenario is then recreated in SUMO, a traffic simulation software, and simulated with the inclusion of vehicles, generating statistics like vehicle coordinates, speed, etc. These results are exported to a CSV file.

The dataset comprises five columns: Vehicle ID, corresponding speed, X and Y positional coordinates, and Lane ID. The number of rows corresponds to the number of vehicles in the simulation.

The dataset is dynamic, as it serves as the basis for constructing a graph. The GNN algorithm is subsequently applied to this graph, generating node embedding for each vehicle.

IMPLEMENTATION

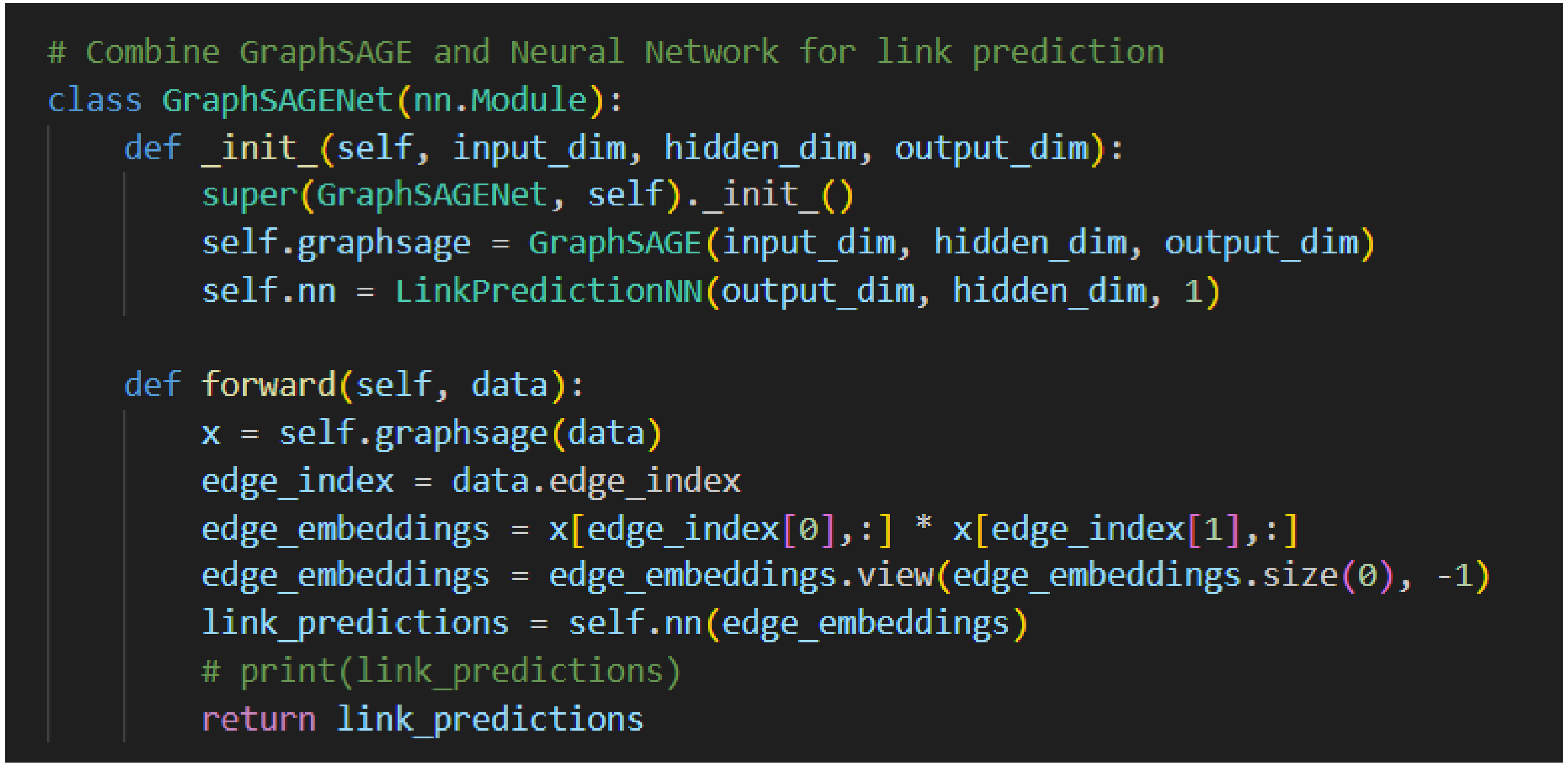
Overall model (GraphSageNET) which gives probabilistic link prediction :

- The proposed approach outlines two neural network modules: a GraphSAGE model for graph-based representation learning and a Link Prediction Neural Network for predicting links between nodes.
- The GraphSAGE model uses graph convolution to generate node embeddings, while the Link Prediction Neural Network uses fully connected layers and rectified linear unit activation to predict link existence.
- Both models have modular and flexible designs, making them suitable for tasks involving graph-based learning and link prediction.

PES UNIVERSITY

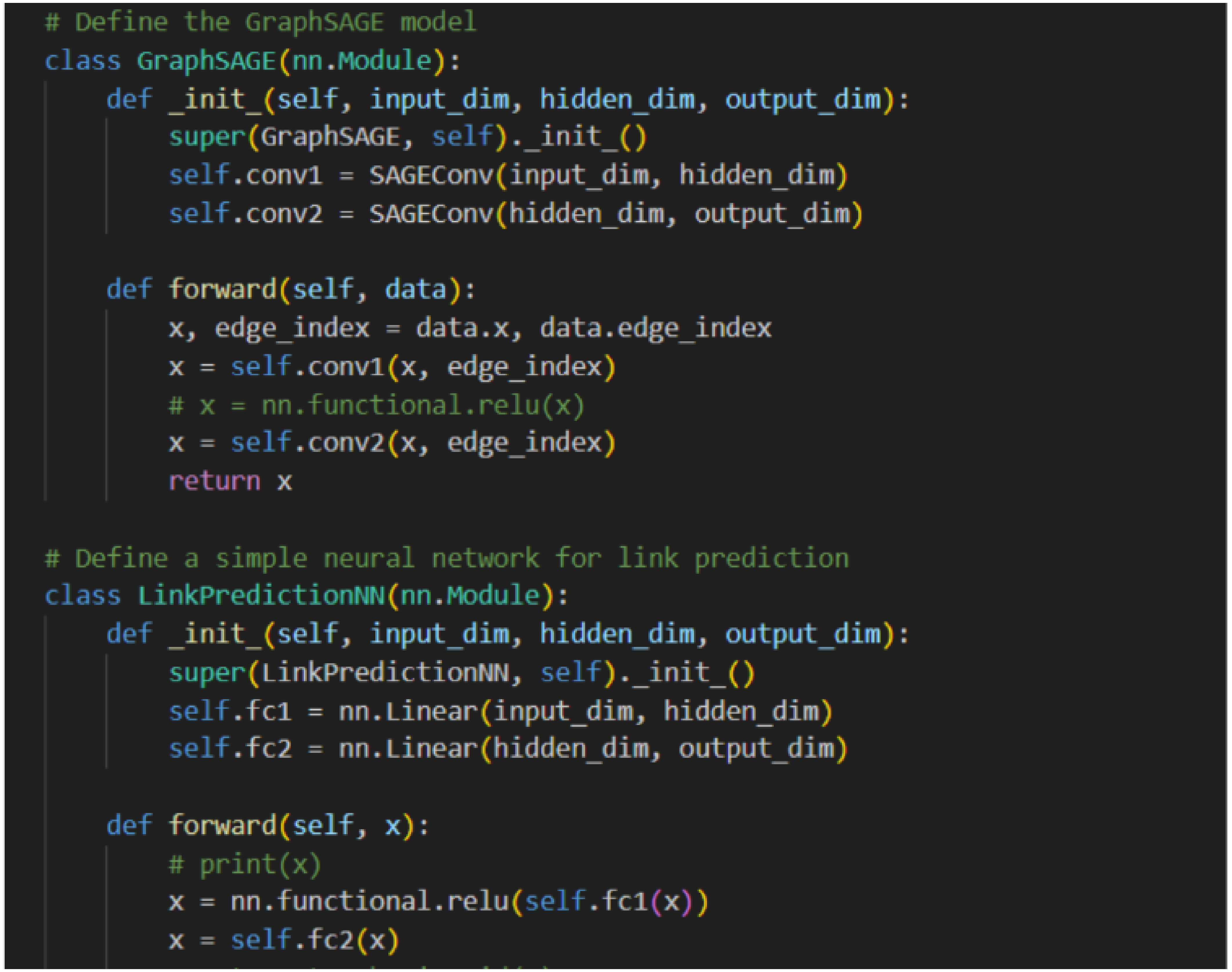
IMPLEMENTATION

Overall model (GraphSageNET) which gives probabilistic link prediction :



```python
# Combine GraphSAGE and Neural Network for link prediction
class GraphSAGENet(nn.Module):
    def _init_(self, input_dim, hidden_dim, output_dim):
        super(GraphSAGENet, self)._init_()
        self.graphsage = GraphSAGE(input_dim, hidden_dim, output_dim)
        self.nn = LinkPredictionNN(output_dim, hidden_dim, 1)

    def forward(self, data):
        x = self.graphsage(data)
        edge_index = data.edge_index
        edge_embeddings = x[edge_index[0],:] * x[edge_index[1],:]
        edge_embeddings = edge_embeddings.view(edge_embeddings.size(0), -1)
        link_predictions = self.nn(edge_embeddings)
        # print(link_predictions)
        return link_predictions
```

GRAPHSAGE , LINK PREDICTOR NEURAL NETWORK :



```python
# Define the GraphSAGE model
class GraphSAGE(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(GraphSAGE, self).__init__()
        self.conv1 = SAGEConv(input_dim, hidden_dim)
        self.conv2 = SAGEConv(hidden_dim, output_dim)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        # x = nn.functional.relu(x)
        x = self.conv2(x, edge_index)
        return x

# Define a simple neural network for link prediction
class LinkPredictionNN(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(LinkPredictionNN, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        # print(x)
        x = nn.functional.relu(self.fc1(x))
        x = self.fc2(x)
```

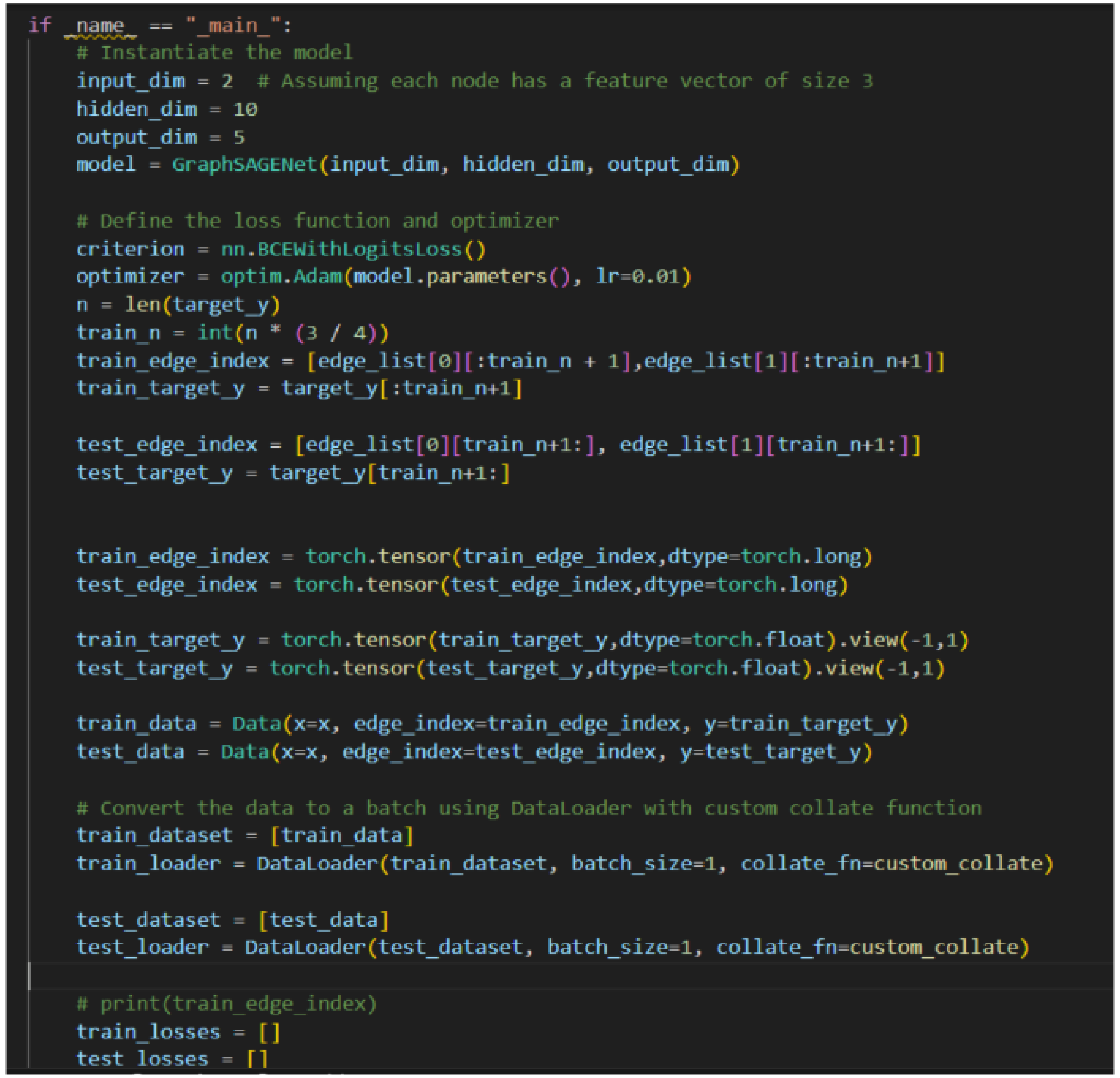
TRAINING THE MODEL ..

Training of the model involves GraphSAGENet model for link prediction in graph-structured data using PyTorch. It involves model instantiation, data preprocessing, training, evaluation, and visualization.

The model is created with specified dimensions for input features, hidden layers, and output dimensions. The loss function and optimizer are set to Binary Cross Entropy with Logits. The dataset is split into training and test sets, with 75% used for training.

Data is formatted into PyTorch Geometric Data objects, and the model is trained over 100 epochs. The model's state dictionary is saved to a file.

TRAINING THE MODEL ..



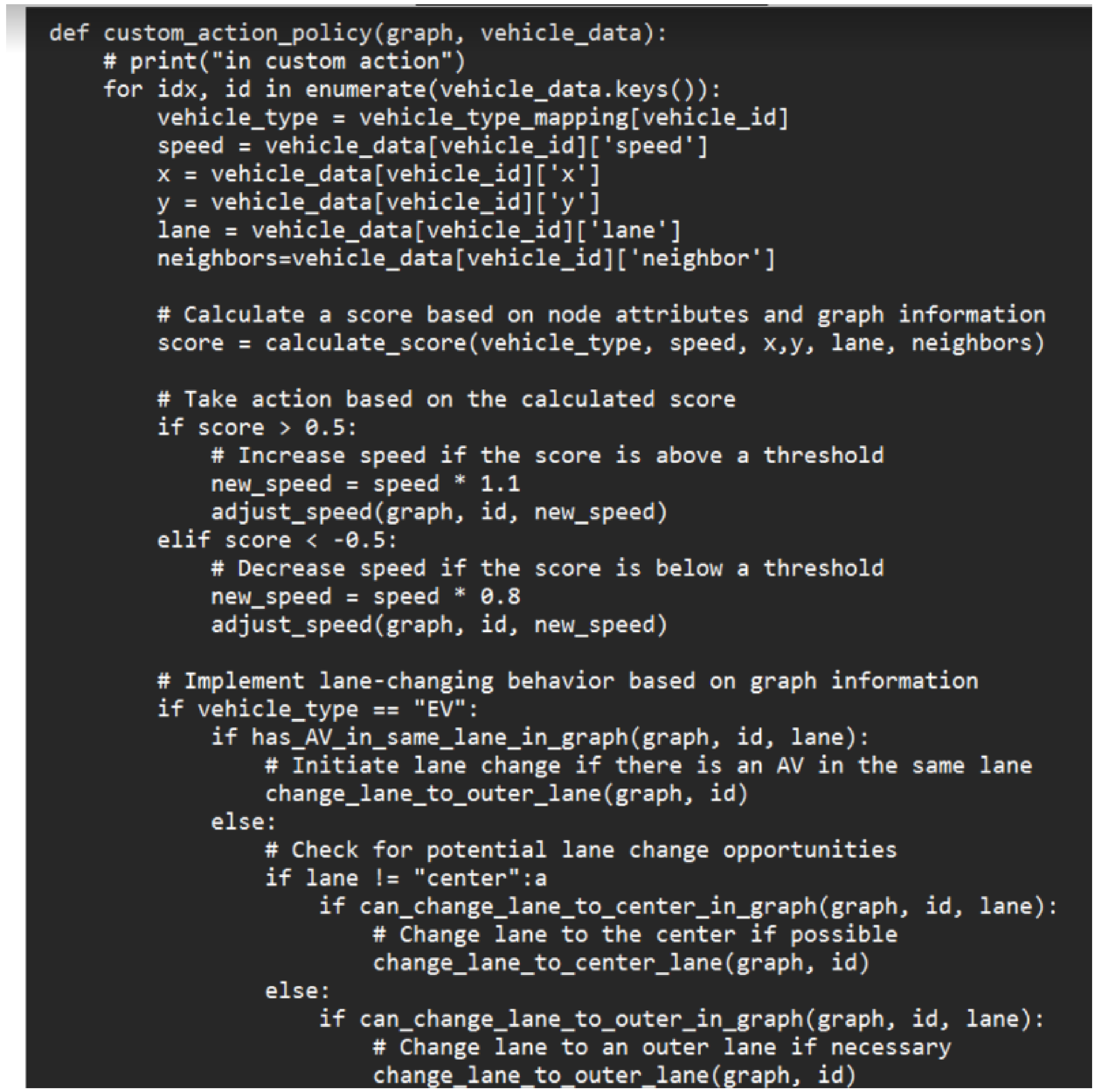
```
if _name_ == "_main_":
    # Instantiate the model
    input_dim = 2  # Assuming each node has a feature vector of size 3
    hidden_dim = 10
    output_dim = 5
    model = GraphSAGENet(input_dim, hidden_dim, output_dim)

    # Define the loss function and optimizer
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.01)
    n = len(target_y)
    train_n = int(n * (3 / 4))
    train_edge_index = [edge_list[0][:train_n + 1],edge_list[1][:train_n+1]]
    train_target_y = target_y[:train_n+1]

    test_edge_index = [edge_list[0][train_n+1:], edge_list[1][train_n+1:]]
    test_target_y = target_y[train_n+1:]


    train_edge_index = torch.tensor(train_edge_index,dtype=torch.long)
    test_edge_index = torch.tensor(test_edge_index,dtype=torch.long)

    train_target_y = torch.tensor(train_target_y,dtype=torch.float).view(-1,1)
    test_target_y = torch.tensor(test_target_y,dtype=torch.float).view(-1,1)

    train_data = Data(x=x, edge_index=train_edge_index, y=train_target_y)
    test_data = Data(x=x, edge_index=test_edge_index, y=test_target_y)

    # Convert the data to a batch using DataLoader with custom collate function
    train_dataset = [train_data]
    train_loader = DataLoader(train_dataset, batch_size=1, collate_fn=custom_collate)

    test_dataset = [test_data]
    test_loader = DataLoader(test_dataset, batch_size=1, collate_fn=custom_collate)

    # print(train_edge_index)
    train_losses = []
    test_losses = []
```

CUSTOM ACTION POLICY :

- This function governs vehicle behavior in a simulation scenario using graph-based representation and individual vehicle data. It extracts information like vehicle type, speed, position, lane, and neighbors from the provided data.
- The function calculates a score for each vehicle based on its attributes and the graph's information, determining whether to adjust speed or change lanes. For emergency vehicles (EVs), the function evaluates the graph structure to initiate lane changes.
- The custom action policy promotes effective and adaptive vehicle navigation in mixed traffic environments.

PES UNIVERSITY

CUSTOM ACTION POLICY :



```
def custom_action_policy(graph, vehicle_data):
    # print("in custom action")
    for idx, id in enumerate(vehicle_data.keys()):
        vehicle_type = vehicle_type_mapping[vehicle_id]
        speed = vehicle_data[vehicle_id]['speed']
        x = vehicle_data[vehicle_id]['x']
        y = vehicle_data[vehicle_id]['y']
        lane = vehicle_data[vehicle_id]['lane']
        neighbors=vehicle_data[vehicle_id]['neighbor']

        # Calculate a score based on node attributes and graph information
        score = calculate_score(vehicle_type, speed, x,y, lane, neighbors)

        # Take action based on the calculated score
        if score > 0.5:
            # Increase speed if the score is above a threshold
            new_speed = speed * 1.1
            adjust_speed(graph, id, new_speed)
        elif score < -0.5:
            # Decrease speed if the score is below a threshold
            new_speed = speed * 0.8
            adjust_speed(graph, id, new_speed)

        # Implement lane-changing behavior based on graph information
        if vehicle_type == "EV":
            if has_AV_in_same_lane_in_graph(graph, id, lane):
                # Initiate lane change if there is an AV in the same lane
                change_lane_to_outer_lane(graph, id)
            else:
                # Check for potential lane change opportunities
                if lane != "center":a
                    if can_change_lane_to_center_in_graph(graph, id, lane):
                        # Change lane to the center if possible
                        change_lane_to_center_lane(graph, id)
                else:
                    if can_change_lane_to_outer_in_graph(graph, id, lane):
                        # Change lane to an outer lane if necessary
                        change_lane_to_outer_lane(graph, id)
```

PES UNIVERSITY

RESULTS

Proposed method diverges from traditional Reinforcement Learning approaches by eliminating the need for high GPU processing power. It achieves action selection without extensive computations

study evaluates the methodology across three simulation scenarios: merge, intersection, and city network. Each scenario is examined under three conditions: baseline, graph construction with the custom action policy function, and GraphSAGE-based graph construction.

The train-test curve for GraphSAGE illustrates low loss values, highlighting the model's effectiveness in capturing and predicting link states in traffic graphs.

INSIGHTS

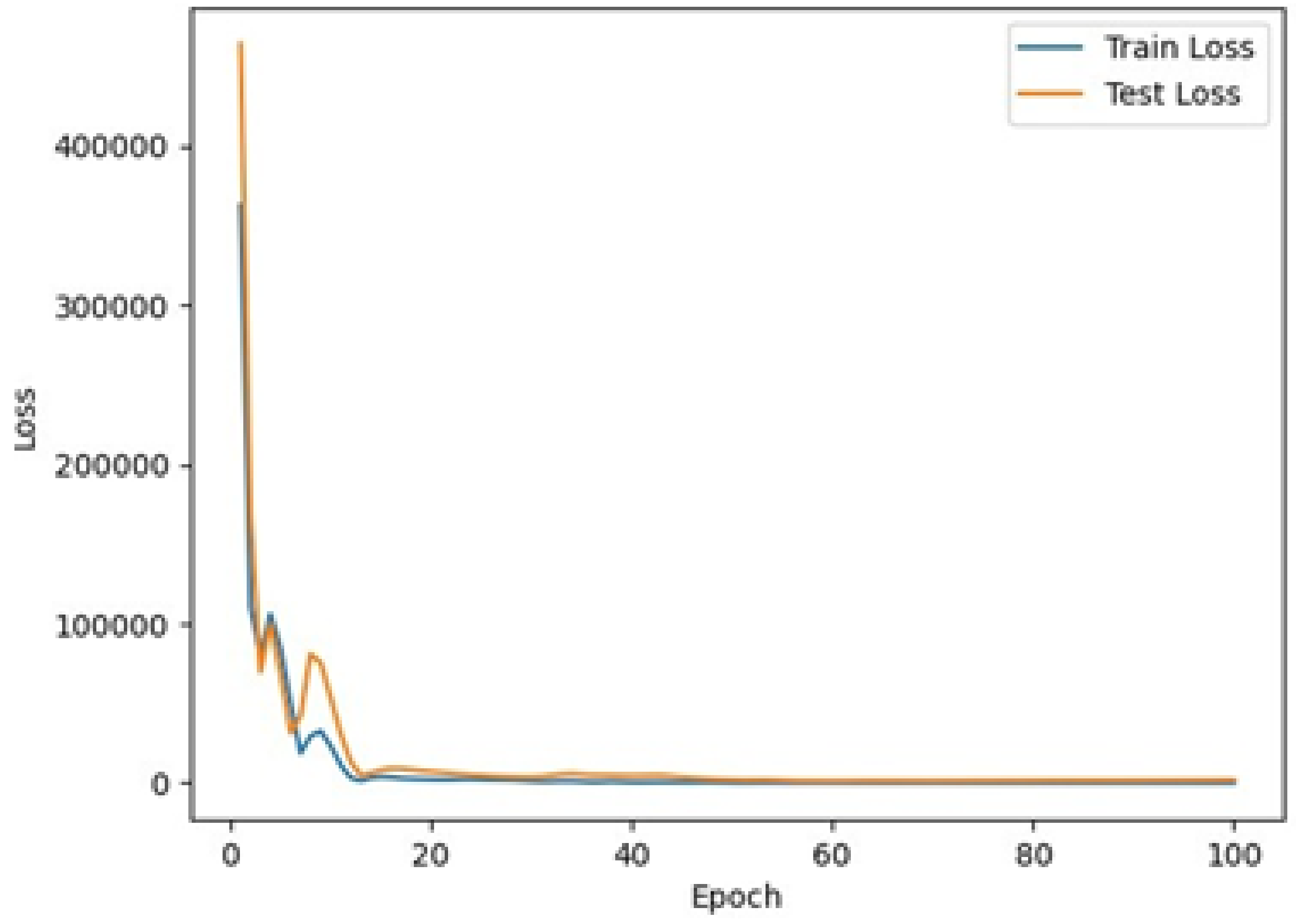


Fig. 4. Train Test curve for GraphSAGE

TABLE III. PERFORMANCE ON INTERSECTION

Simulation Scenario	**Parameters**		
	Emergency Waiting Time	***Depart Delay***	***Average Waiting Time***
Baseline	9.2s	2.4s	15.8s
Graph made by Simulation attributes	3.5s	0.2s	3.2s
Graph made by GraphSAGE	3.8s	0.5s	3.1s

RESULTS

In the merge scenario, the performance improvement over the baseline is notable, . The GraphSAGE-based approach demonstrates performance close to that of the simulation-based approach, emphasizing GraphSAGE's ability to capture traffic dynamics

optimization effect in the intersection scenario is more pronounced than in the merge scenario. The increased mutual influence between cars in intersections contributes to a significant performance boost

custom action policy function, combined with the graph for the city network, significantly outperforms the baseline. The GraphSAGE-based approach for graph construction aligns closely with the simulation-based approach

LESSON LEARNT

Through the proposed research work, application of Link prediction neural networks in real life scenarios play a vital role, although many authors have tried with application of RL algorithms in efficient traversal of set of vehicles in an environment, few to none have focused on single agent traversal , Emergency vehicle in particular.

Through the Link Predictor module, In real life scenario, when there is a case of sensor failure , our module can effectively judge the new links possible so that emergency vehicle wont loose its connection and reach destination quickly

By Comparing with multiple baselines ,for each of the scenario it is observed that our model outperforms baseline models in case of Intersection scenario with better metrics showcasing less depart delay and waiting time.

PES UNIVERSITY

CONCLUSION

This novel approach to emergency vehicle (EV) navigation in mixed-traffic environments by combining GraphSAGE for graph representation and link prediction with a custom action policy function. This integration captures the dynamic and unpredictable nature of traffic, outperforming baseline methods

suggested method offers several advantages over conventional Reinforcement Learning (RL) approaches. GraphSAGE enables more precise link prediction by capturing intricate spatial relationships in the traffic network. The custom action policy function optimizes traffic flow for EVs, resulting in more effective navigation.

Project Github Link (with report,paper,poster,video):

https://github.com/ShubhamSBhat/Capstone

PES UNIVERSITY

FUTURE SCOPE

The research highlights the potential of graph-based traffic modeling and prediction methods, especially in crucial applications like EV navigation.

The suggested methodology provides a viable means to enhance EV navigation effectiveness and ensure prompt assistance during emergencies.

Future research directions include exploring various graph neural network architectures, developing a more complex action policy function considering additional variables, and testing the method on a larger and more diverse dataset to improve efficacy and applicability in real-world traffic scenarios.

PES UNIVERSITY

REFERENCES

- Liu, Qi, Zirui Li, Xueyuan Li, Jingda Wu, and Shihua Yuan. "Graph Convolution-Based Deep Reinforcement Learning for Multi-Agent Decision-Making in Mixed Traffic Environments." arXiv preprint arXiv:2201.12776 (2022).
- Sun, Gang, Yijing Zhang, Dan Liao, Hongfang Yu, Xiaojiang Du, and Mohsen Guizani. "Bus-trajectory-based street-centric routing for message delivery in urban vehicular ad hoc networks." IEEE Transactions on Vehicular Technology 67, no. 8 (2018): 7550-7563.
- Dong, Jiqian, Sikai Chen, Paul Young Joun Ha, Yujie Li, and Samuel Labi. "A drl-based multiagent cooperative control framework for cav networks: a graphic convolution q network." arXiv preprint arXiv:2010.05437 (2020).
- Shi, Tianyu, Jiawei Wang, Yuankai Wu, Luis Miranda-Moreno, and Lijun Sun. "Efficient connected and automated driving system with multi-agent graph reinforcement learning." arXiv:2007.02794 (2020).

THANK YOU

PES
UNIVERSITY