Import traci

Import random

From agent import DQNAgent

CONFIG\_FILE = “cross3ltl.sumocfg”

Edges = [“1fi”, “2fi”, “3fi”, “4fi”] # Edges to track

Lanes = [“1fi\_0”, “2fi\_0”, “3fi\_0”, “4fi\_0”]

Def get\_state():

Cars = [traci.lane.getLastStepVehicleNumber(lane) for lane in lanes]

Current\_phase = traci.trafficlight.getPhase(“0”)

Return cars + [current\_phase]

MIN\_GREEN = 10 # min steps to hold green if cars exist

Def run():

Agent = DQNAgent(state\_size=5, action\_size=2)

Traci.start([“sumo-gui”, “-c”, CONFIG\_FILE])

Prev\_wait = sum(traci.lane.getLastStepHaltingNumber(lane) for lane in lanes)

Time\_since\_last\_change = 0

For step in range(200):

Traci.simulationStep()

Time\_since\_last\_change += 1

# State

State = get\_state()

Action = agent.act(state)

# Check if current green lane has cars

Current\_phase = traci.trafficlight.getPhase(“0”)

Green\_lanes = traci.trafficlight.getControlledLanes(“0”)

Active\_lane = green\_lanes[current\_phase] # lane(s) currently green

Cars\_on\_green = traci.lane.getLastStepVehicleNumber(active\_lane)

# Only allow switch if:

# 1. Min green passed, OR

# 2. No cars on green

If action == 1 and (time\_since\_last\_change >= MIN\_GREEN or cars\_on\_green == 0):

Phases = traci.trafficlight.getCompleteRedYellowGreenDefinition(“0”)[0].phases

Num\_phases = len(phases)

Next\_phase = (current\_phase + 1) % num\_phases

Traci.trafficlight.setPhase(“0”, next\_phase)

Time\_since\_last\_change = 0

# Reward

Total\_stopped = sum(traci.lane.getLastStepHaltingNumber(lane) for lane in lanes)

Reward = prev\_wait – total\_stopped

Prev\_wait = total\_stopped

# Train

Next\_state = get\_state()

Agent.train(state, action, reward, next\_state, done=False)

Print(f”Step {step}: Phase={current\_phase}, CarsOnGreen={cars\_on\_green}, Action={action}, Reward={reward}, Queue={total\_stopped}”)

Agent.model.save(“trained\_model.h5”)

Traci.close()

If \_\_name\_\_ == “\_\_main\_\_”:

Run()