# RWR 4015
# Traffic Simulation for Planning Applications

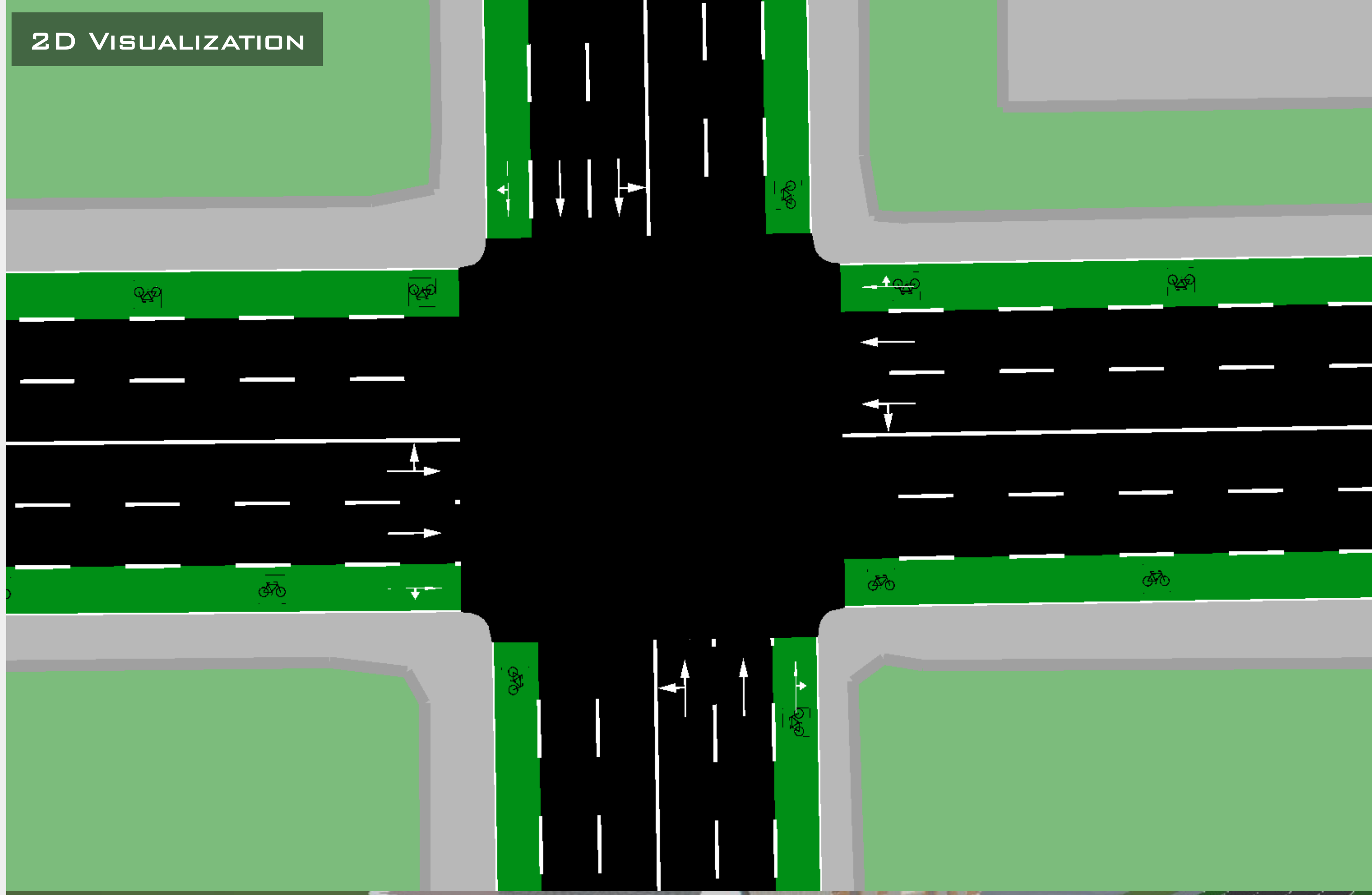
Dr. Ahmad Mohammadi

Week 9 | Hands-on:
Artificial Intelligence in
Intelligent Transportation Systems

Fall 2026

RoadwayVR

roadwayvr.github.io/TrafficSimulationforPlanningApplications

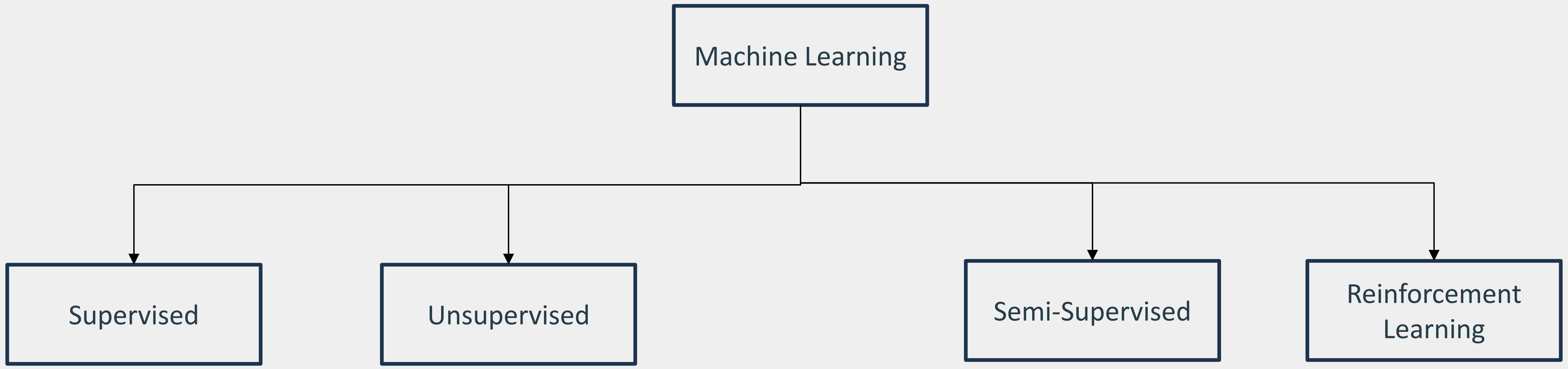



# Agenda

**Goal: Develop an Intelligent Transportation System (Smart Traffic Signal)**

- ❑ Fundamental of Machine Learning Algorithms
- ❑ Fundamental of Reinforcement Learning Algorithms
- ❑ Implement an Intelligent Traffic Signal
- ❑ Analyze the Performance of Intelligent Traffic Signal

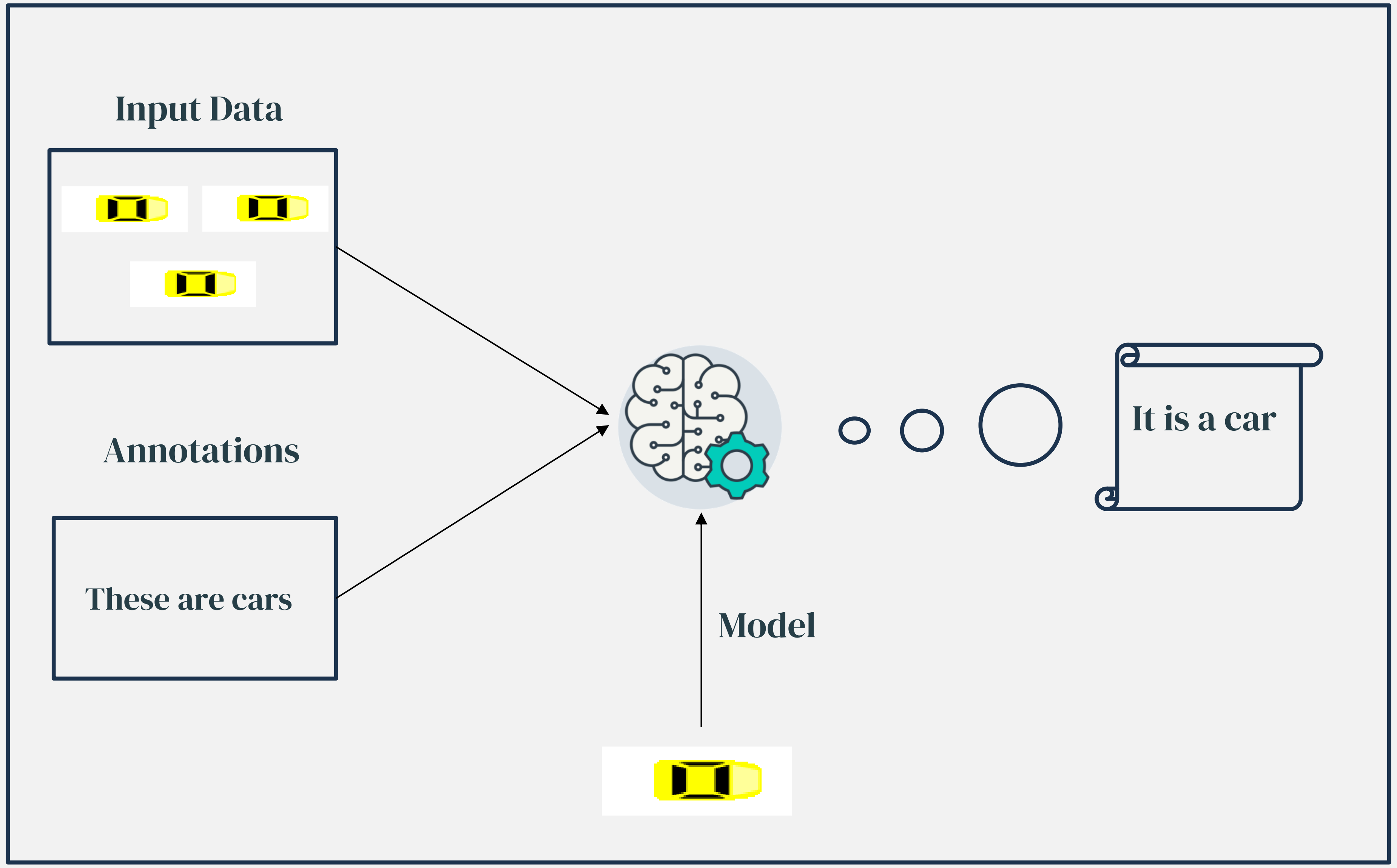
# Fundamental of Machine Learning Algorithms



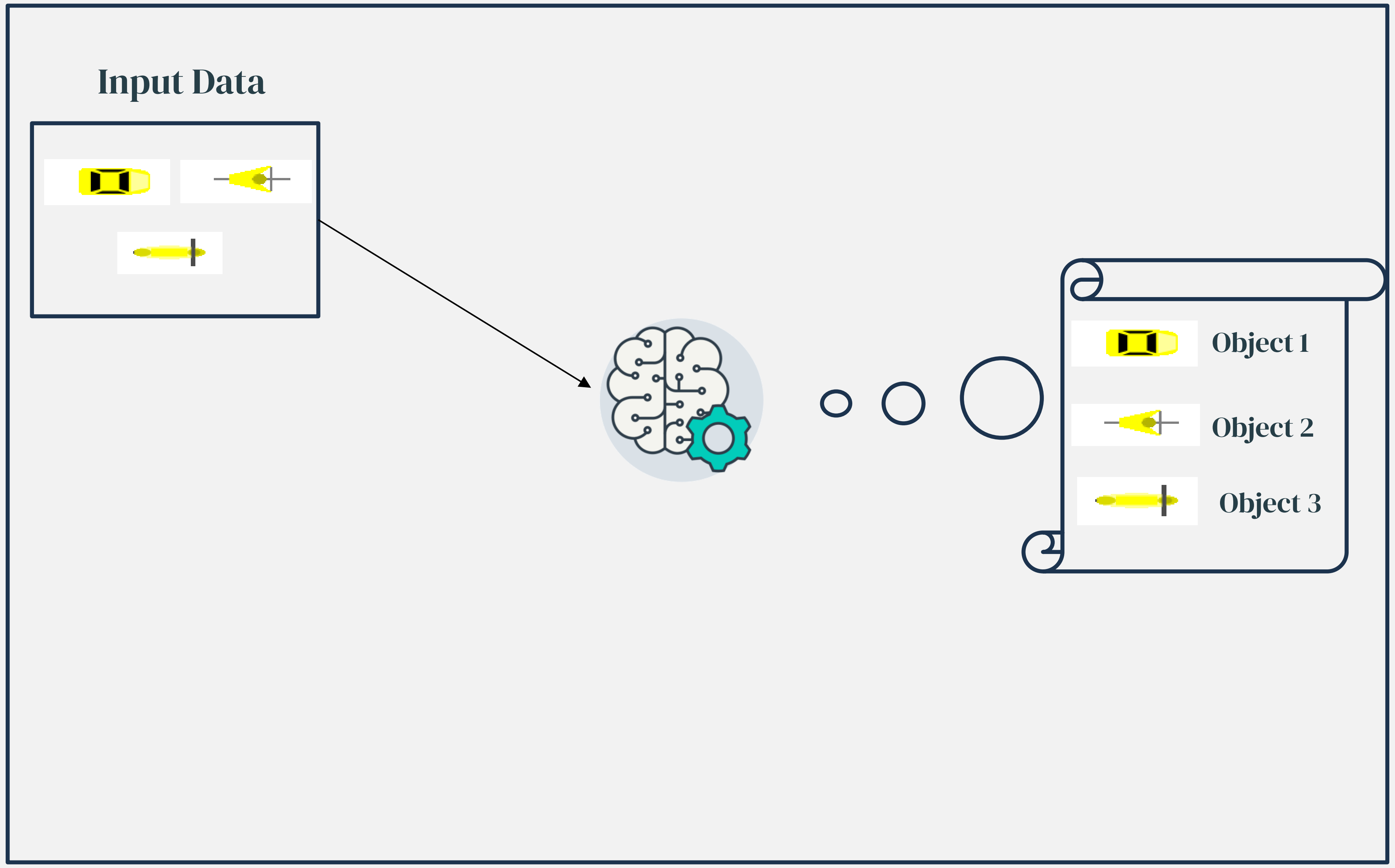

**Reference:** Wu, D., Wang, S., Liu, Q., Abualigah, L., & Jia, H. (2022). An Improved Teaching-Learning-Based Optimization Algorithm with Reinforcement Learning Strategy for Solving Optimization Problems. *Computational Intelligence and Neuroscience*, *2022*(1), 1535957.

# Supervised/Unsupervised Learning

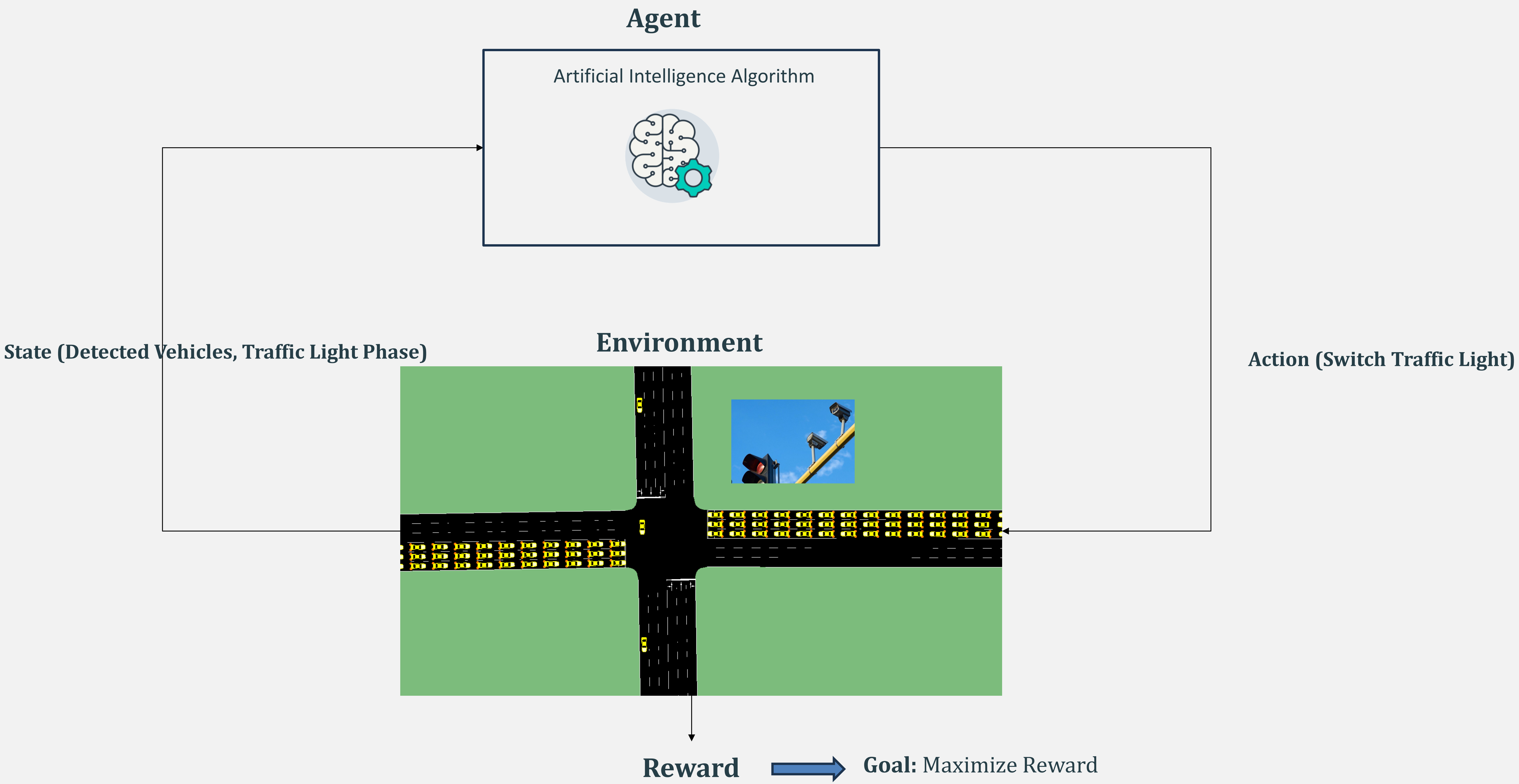
Supervised Learning




Unsupervised Learning



# Reinforcement Learning

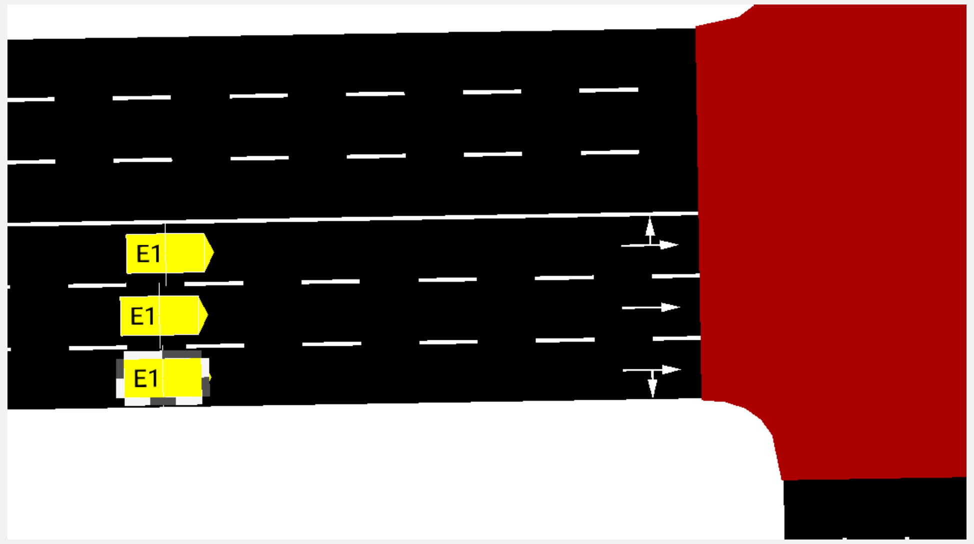


# State: How to Collect Data

### A. Inductive Loop in Real-World

**1. Vehicle count:**
How many vehicles have passed over time.
**2. Vehicle Speed:**
Mean speed from the vehicles passing a specific point.

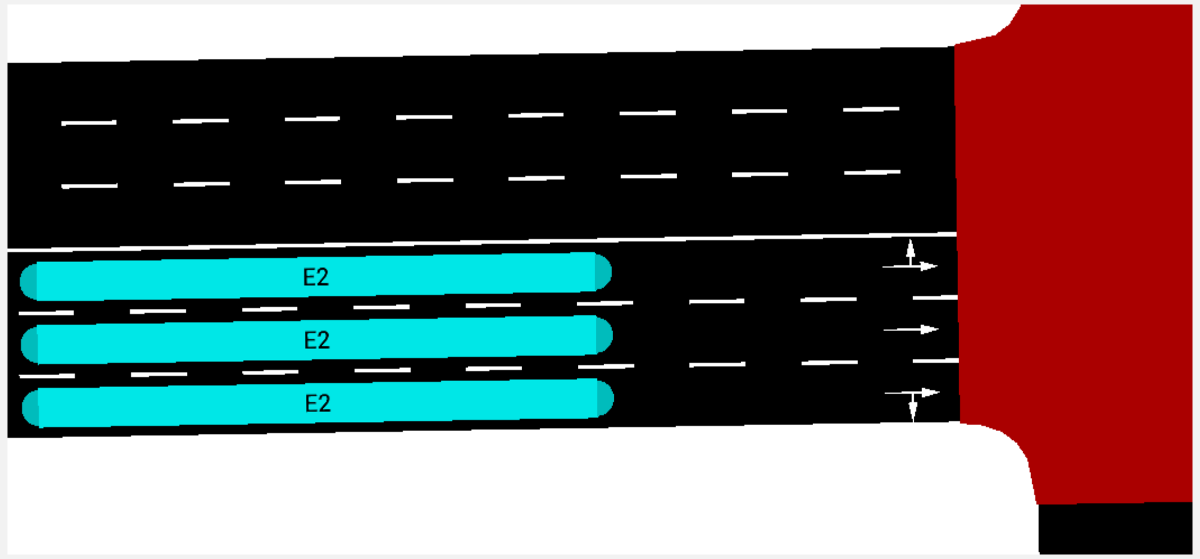
### A. Inductive Loop in SUMO




### B. Camera (Computer Vision) in Real-World

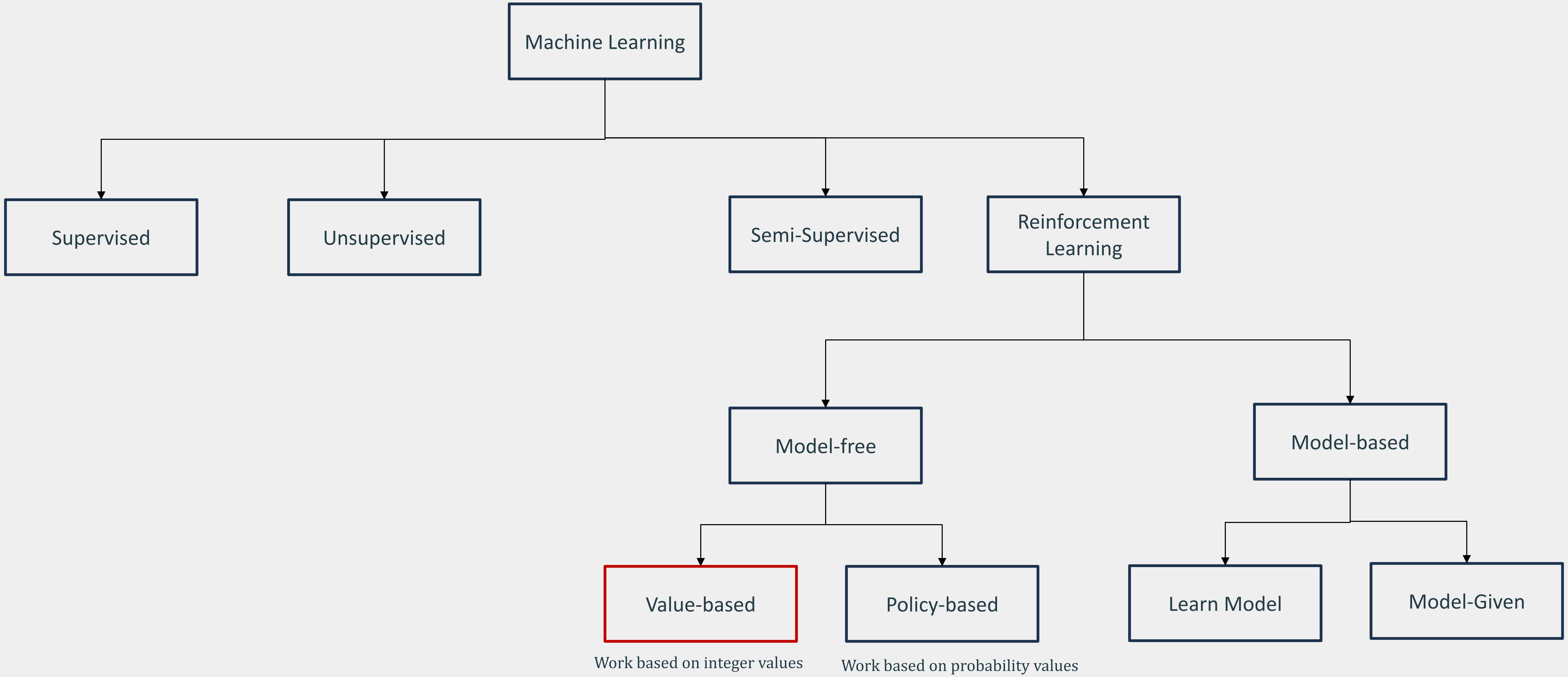
**1. Vehicle count:**
How many vehicles have passed over time.
**2. Vehicle Speed:**
Mean speed from the vehicles passing a specific point.
**3. Queue length:**
How many vehicles are “stacked up” or moving slowly in the detector’s monitored area.
**4. Occupancy :**
The fraction of time that the detector area is occupied by at least one vehicle.

### B. Camera (Computer Vision) in SUMO (Lane Area)



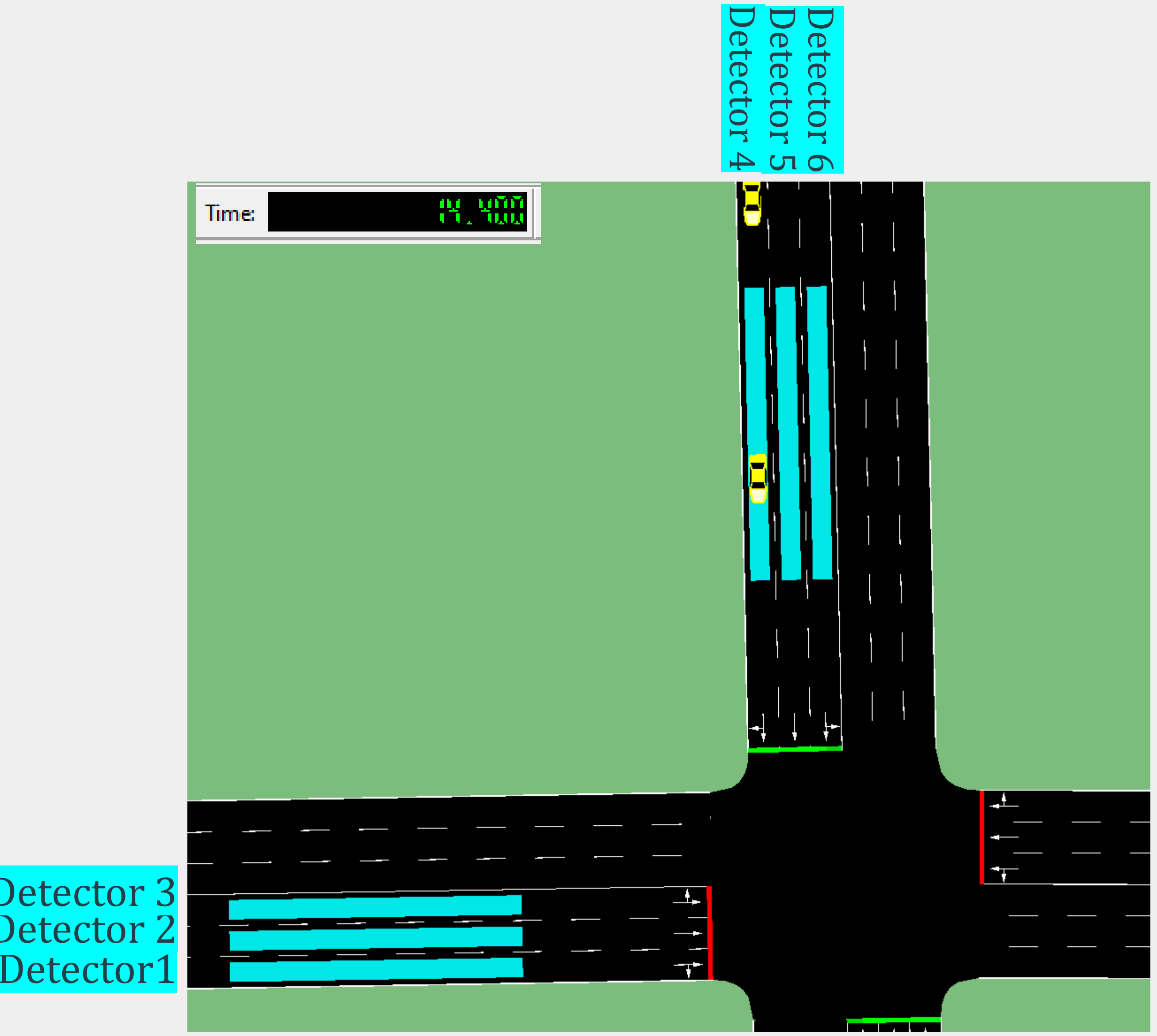

**Reference:** Klein, L. A., Mills, M. K., Gibson, D., & Klein, L. A. (2006). *Traffic detector handbook: Volume II* (No. FHWA-HRT-06-139). United States. Federal Highway Administration.

# Agent: Different Algorithms




**Reference:** Wu, D., Wang, S., Liu, Q., Abualigah, L., & Jia, H. (2022). An Improved Teaching-Learning-Based Optimization Algorithm with Reinforcement Learning Strategy for Solving Optimization Problems. *Computational Intelligence and Neuroscience*, *2022*(1), 1535957.

# Q-learning




- Assume we name detectors (1-6)
- Traffic light has these phases:

| | dur | |
|---|---|---|
| 0 | 42.00 | rrrrrGGGGgrrrrr |
| 1 | 3.00 | rrrrryyyyyrrrrr |
| 2 | 42.00 | GGGGgrrrrrGGGGg |
| 3 | 3.00 | yyyyyrrrrryyyyy |

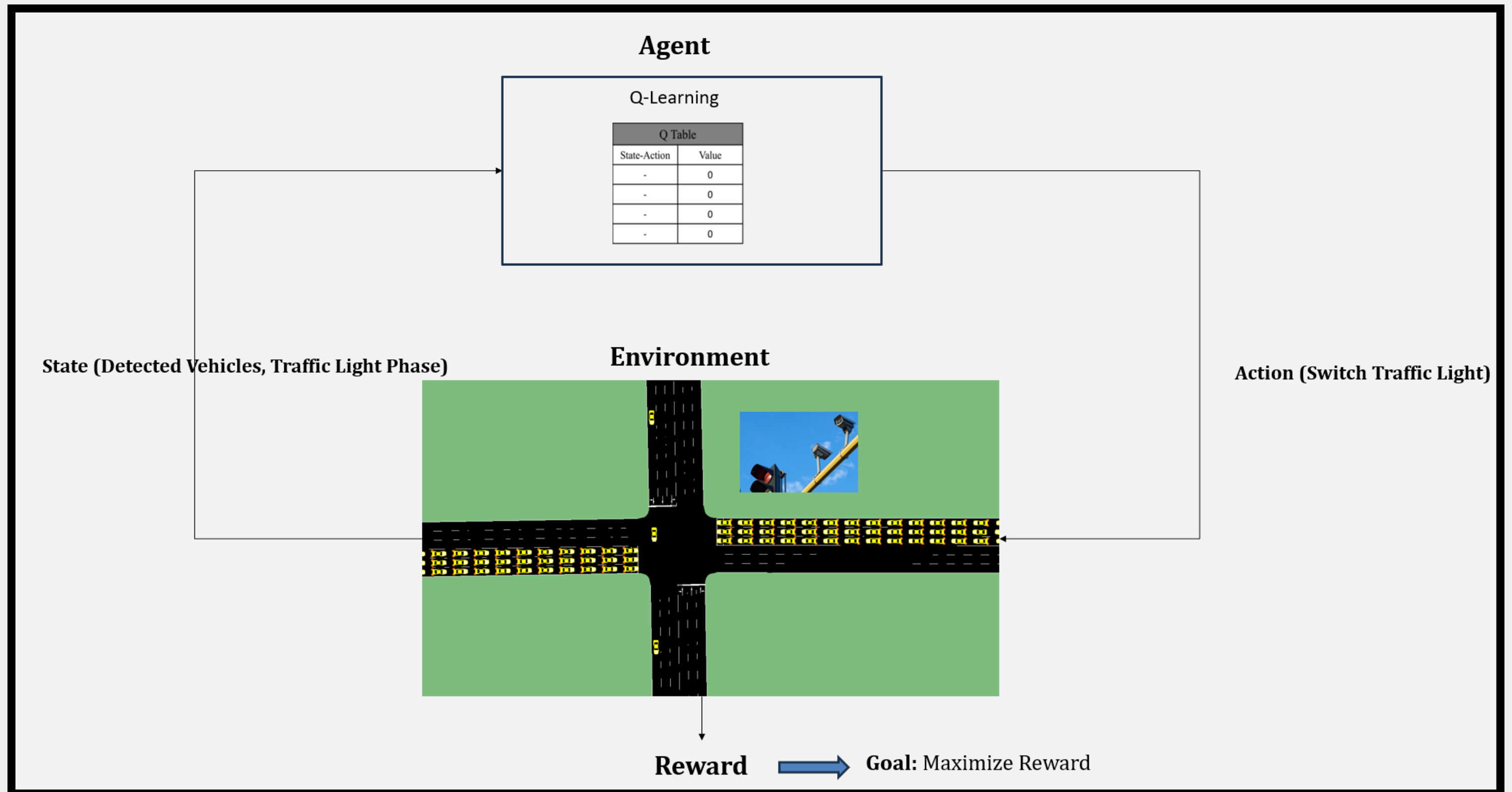
- Action 0: Keep Phase
- Action 1: Switch Phase

**Q1: Whats Current State?**
**Answer: (0,0,0,1,0,0,0)**

**Q2: Whats the <u>likely</u> Action (consider there is no Q-table)?**
**Answer: 0**

**Q3: Whats Next State?**
**Answer: (0,0,0,1,0,0,0)**

**Q4: Whats Reward?**
**Answer: -1**



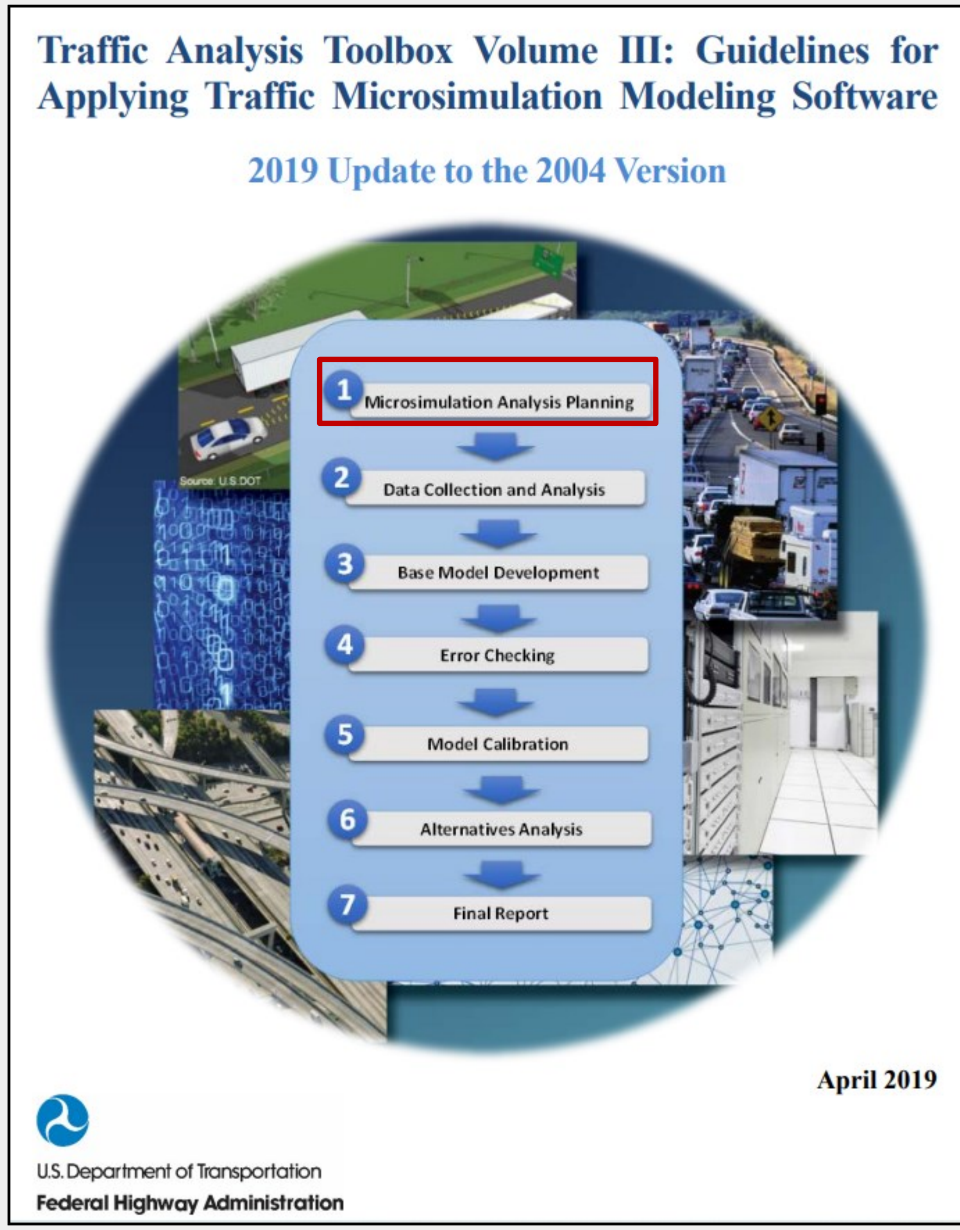
# Step 1 Micro-Simulation Analysis Planning

***Goal and Objectives:***

- ❑ Developing Two Reinforcement Learning Algorithms (Q-Learning and Deep Q-Learning) for Traffic Signal Control
- ❑ Compare Results with Static Traffic Signal

***Study Area:***

- ❑ A Random Intersection (For Tutorial Only)




Reference link:
https://ops.fhwa.dot.gov/publications/fhwahop18036/fhwahop18036.pdf

# Step 2 Data Collection and Analysis

**Required Data:**

❑ **Traffic Flow Parameters:**

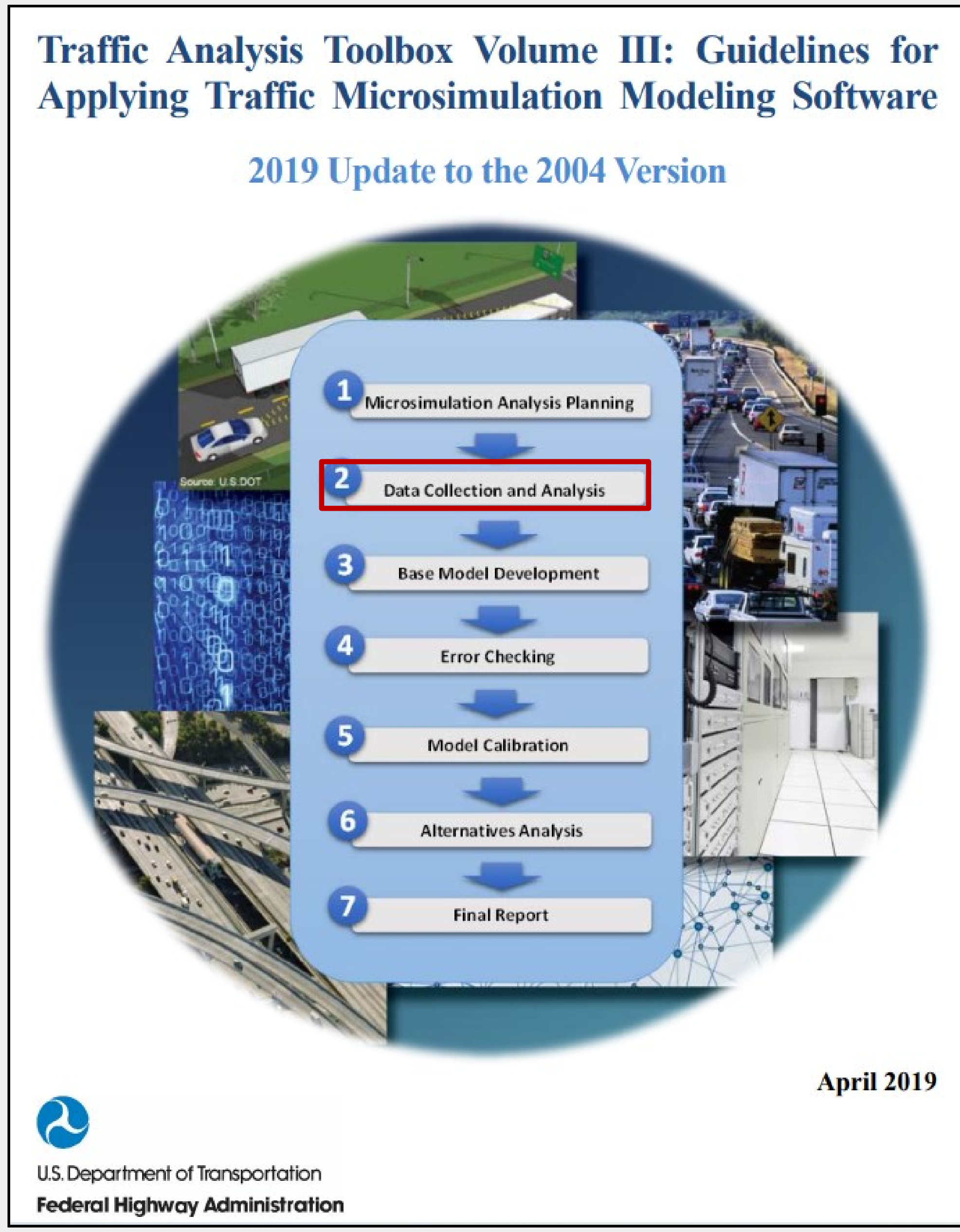
A. **Traffic Volume:** Default SUMO Values
B. **Traffic Speed:** Default SUMO Values

❑ **Traffic Behavioral Parameters:**

A. **Lane Changing Model:** Default SUMO Values
B. **Car Following Model:** Default SUMO Values

❑ **Vehicle Types:**

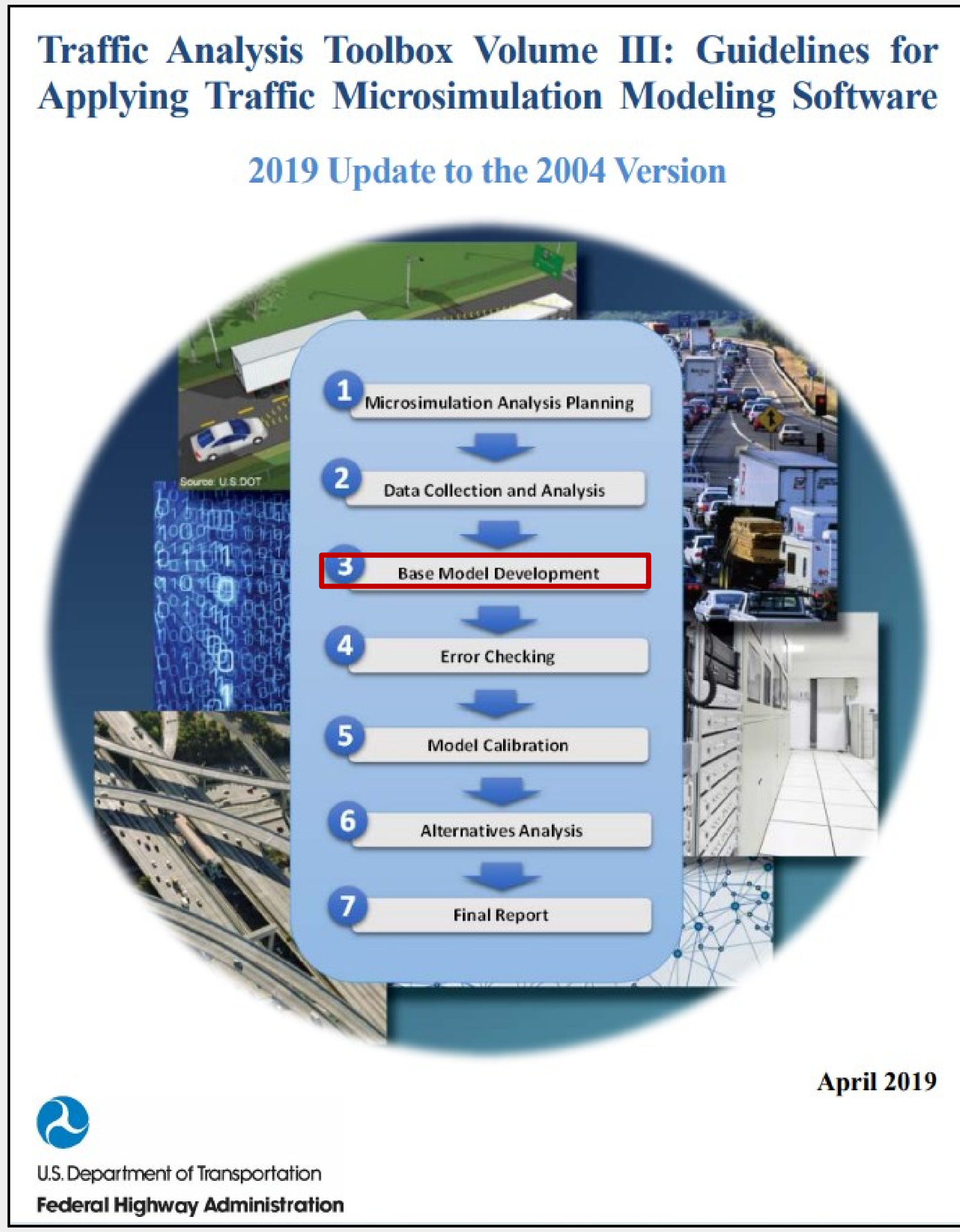
A. **Default SUMO Values**



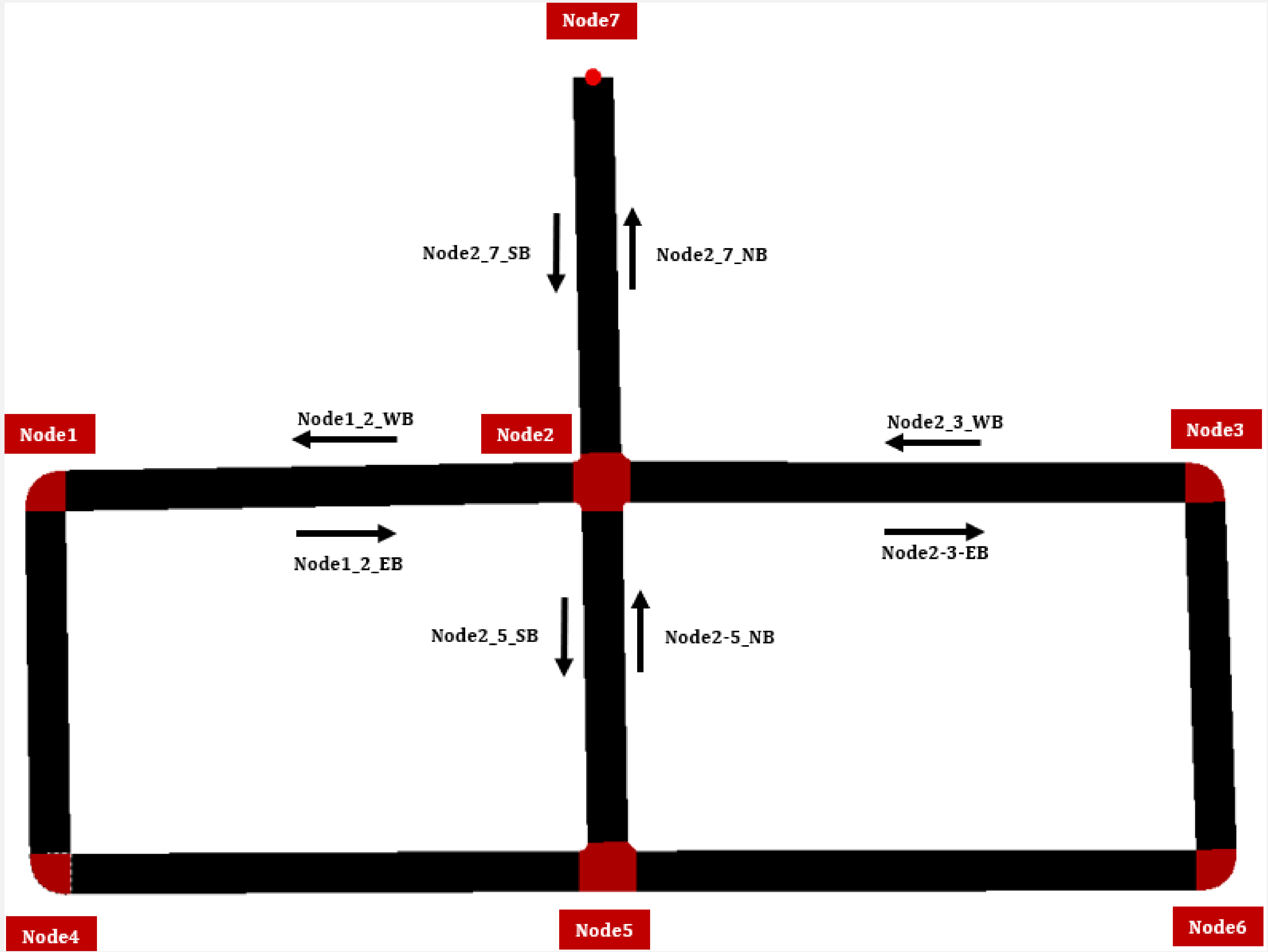
# Step 3 Base Model Development

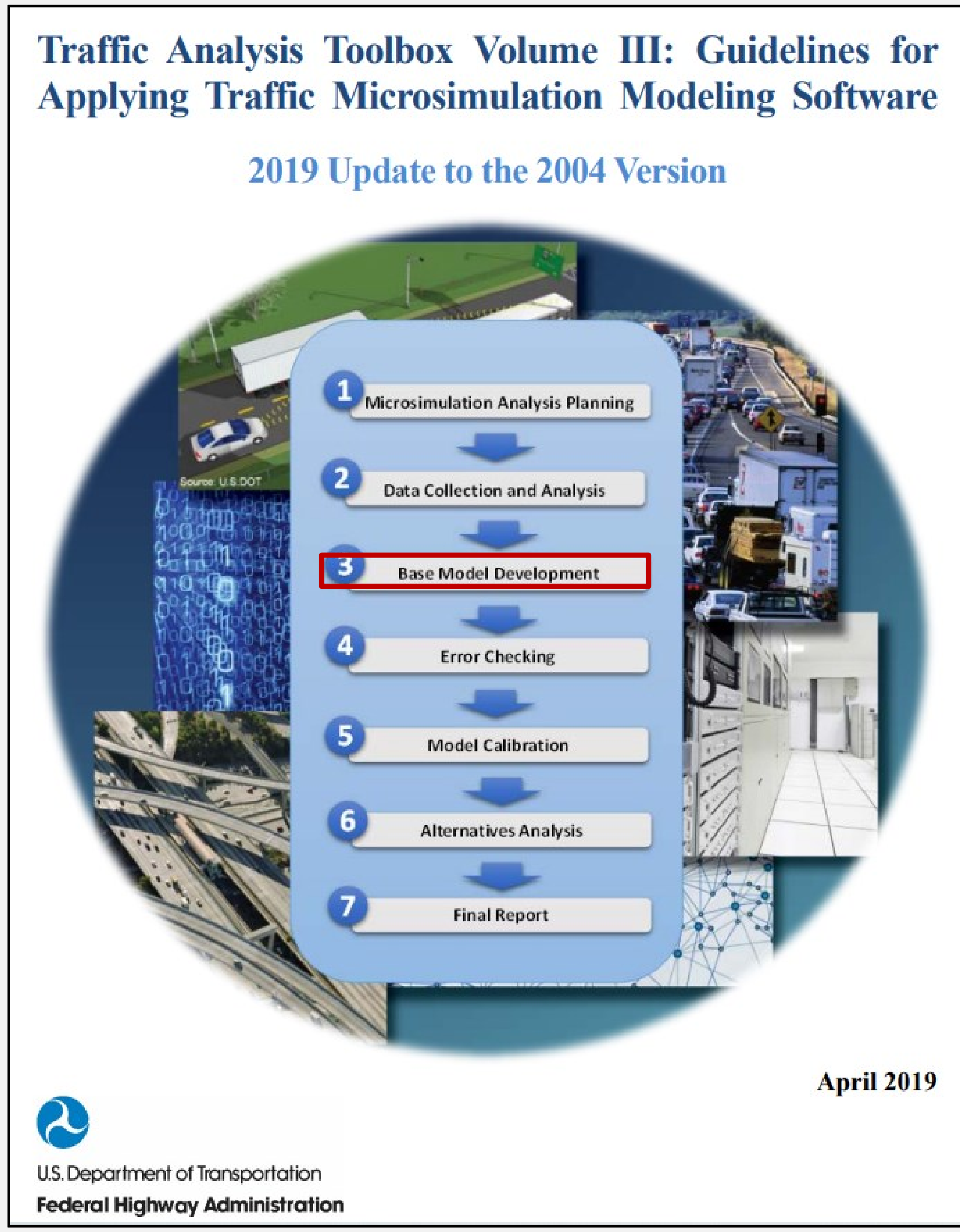
**Required Steps:**

A. Create Network and Name it Properly
B. Adding Traffic Cars, Volume and Speed
C. Vehicle Types
D. Adding Detector
E. Add Traffic Light: 1. Fixed Timing 2. Q-Learning 3- Deep Q Learning



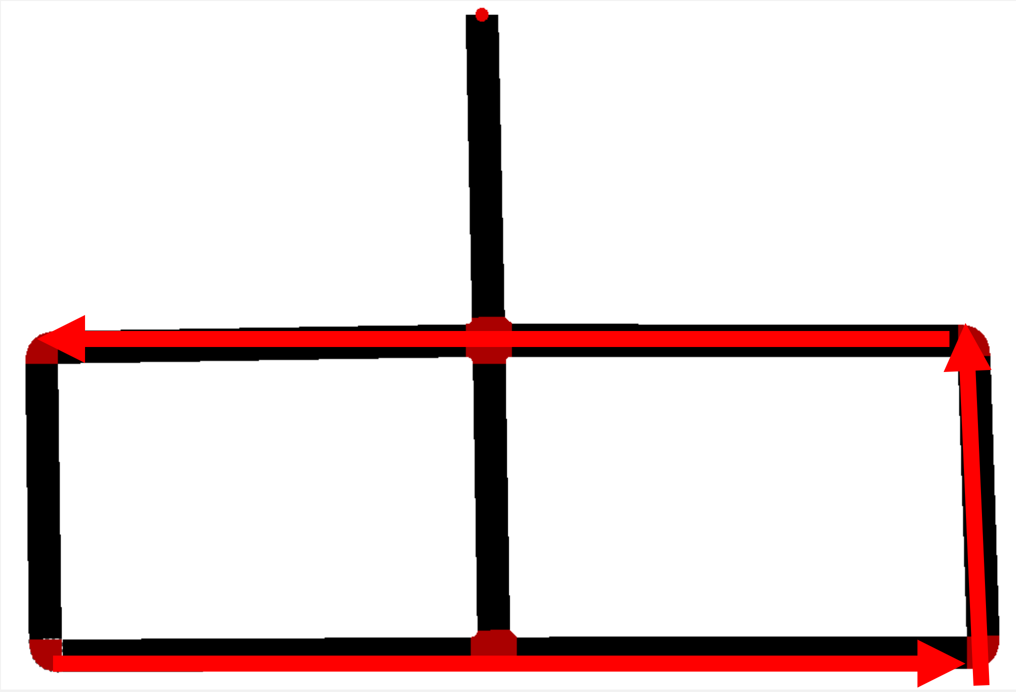
# A. Create Network and Name it Properly



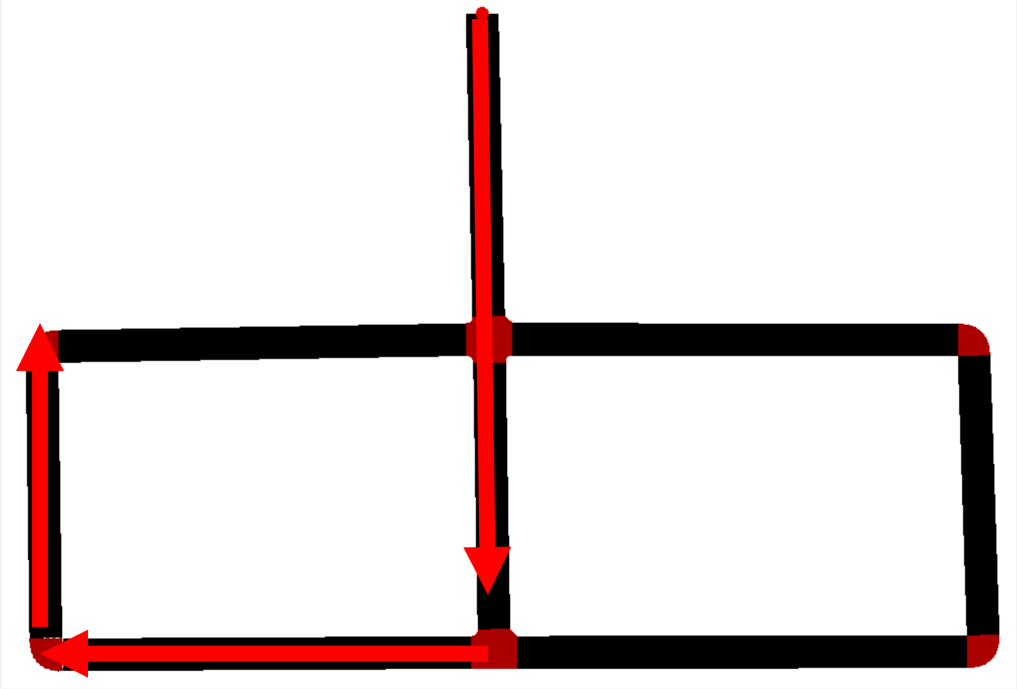



# B. Adding Traffic Cars, Volume and Speed

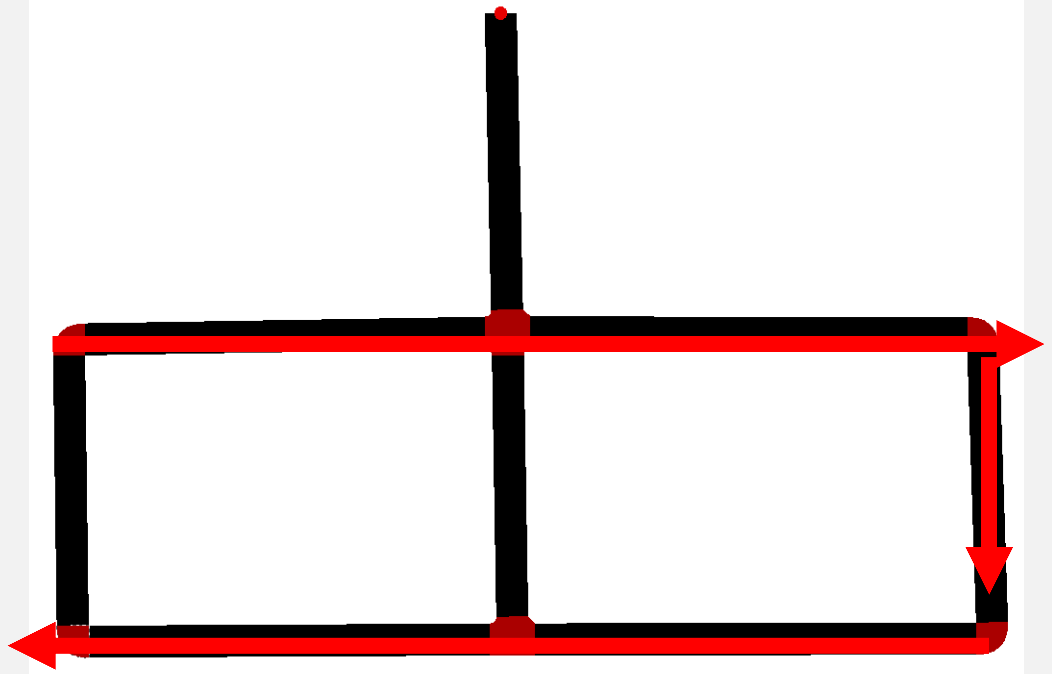
F_0: 1800



F_1: 1800

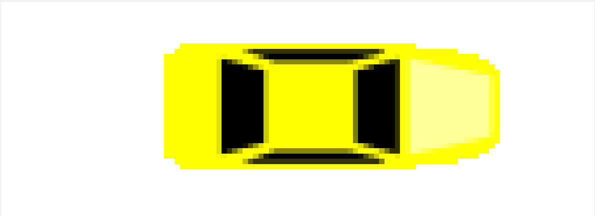


F_2: 1800



# C. Vehicle Types

Default



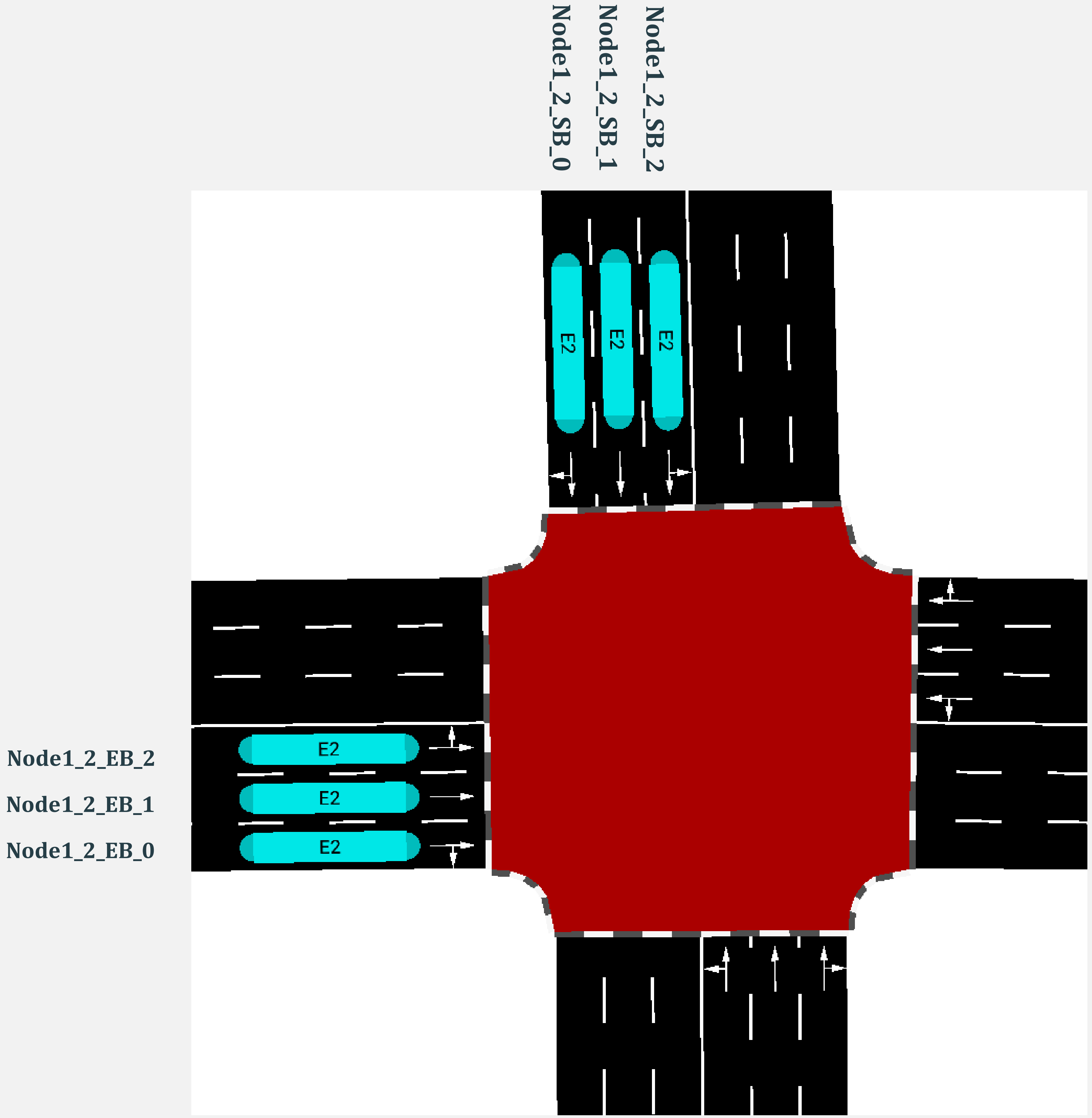
# D. Adding Detectors

## 1. Camera (Computer Vision) in SUMO (Lane Area)

| Variable | ValueType | Description | Python Method |
|---|---|---|---|
| id list (0x00) | stringList | Returns a list of ids of all lane area detectors within the scenario (the given DetectorID is ignored) | getIDList |
| count (0x01) | int | Returns the number of lane area detectors within the scenario (the given DetectorID is ignored) | getIDCount |
| position (0x42) | double | Returns the starting position of the detector at its lane, counted from the lane's begin, in meters. | getPosition |
| length(0x44) | double | Returns the length of the detector in meters. | getLength |
| lane ID (0x51) | string | Returns the ID of the lane the detector is placed at. | getLaneID |
| last step vehicle number (0x10) | int | Returns the number of vehicles that have been within the area detector within the last simulation step [#]; | getLastStepVehicleNumber |
| last step mean speed (0x11) | double | Returns the mean speed of vehicles that have been within the named area detector within the last simulation step [m/s] | getLastStepMeanSpeed |
| last step vehicle ids (0x12) | stringList | Returns the list of ids of vehicles that have been within the detector in the last simulation step | getLastStepVehicleIDs |
| last step occupancy (0x13) | int | Returns the percentage of space the detector was occupied by a vehicle [%] | getLastStepOccupancy |
| last step halting vehicles number (0x14) | int | Returns the number of vehicles which were halting during the last time step | getJamLengthVehicle |
| last step jam length in number of vehicles (0x18) | int | Returns the number of vehicles which were halting on the loop during the last time step | getJamLengthVehicle |
| last step jam length in meters (0x19) | int | Returns the length of the jam in meters | getJamLengthMeters |
| interval occupancy (0x23) | double | The average percentage of the detector length that was occupied by a vehicle during the current interval | getIntervalOccupancy |
| interval speed (0x24) | double | The average (time mean) speed of vehicles during the current interval | getIntervalMeanSpeed |
| interval number (0x25) | int | The number of vehicles (or persons, if so configured) that passed the detector during the current interval | getIntervalVehicleNumber |
| interval max jam length in meters (0x32) | stringList | The maximum jam length in meters during the current interval | getIntervalMaxJamLengthInMeters |
| last interval occupancy (0x27) | double | The average percentage of the detector length that was occupied by a vehicle during the previous interval | getLastIntervalOccupancy |
| last interval speed (0x28) | double | The average (time mean) speed of vehicles during the previous interval | getLastIntervalMeanSpeed |
| last interval number (0x29) | int | The number of vehicles (or persons, if so configured) that passed the detector during the previous interval | getLastIntervalVehicleNumber |
| last interval max jam length in meters (0x33) | stringList | TThe maximum jam length in meters during the previous interval | getLastIntervalMaxJamLengthInMeters |

https://sumo.dlr.de/docs/Simulation/Output/Lanearea_Detectors_%28E2%29.html

# D. Adding Detectors



# E. Adding Traffic Light

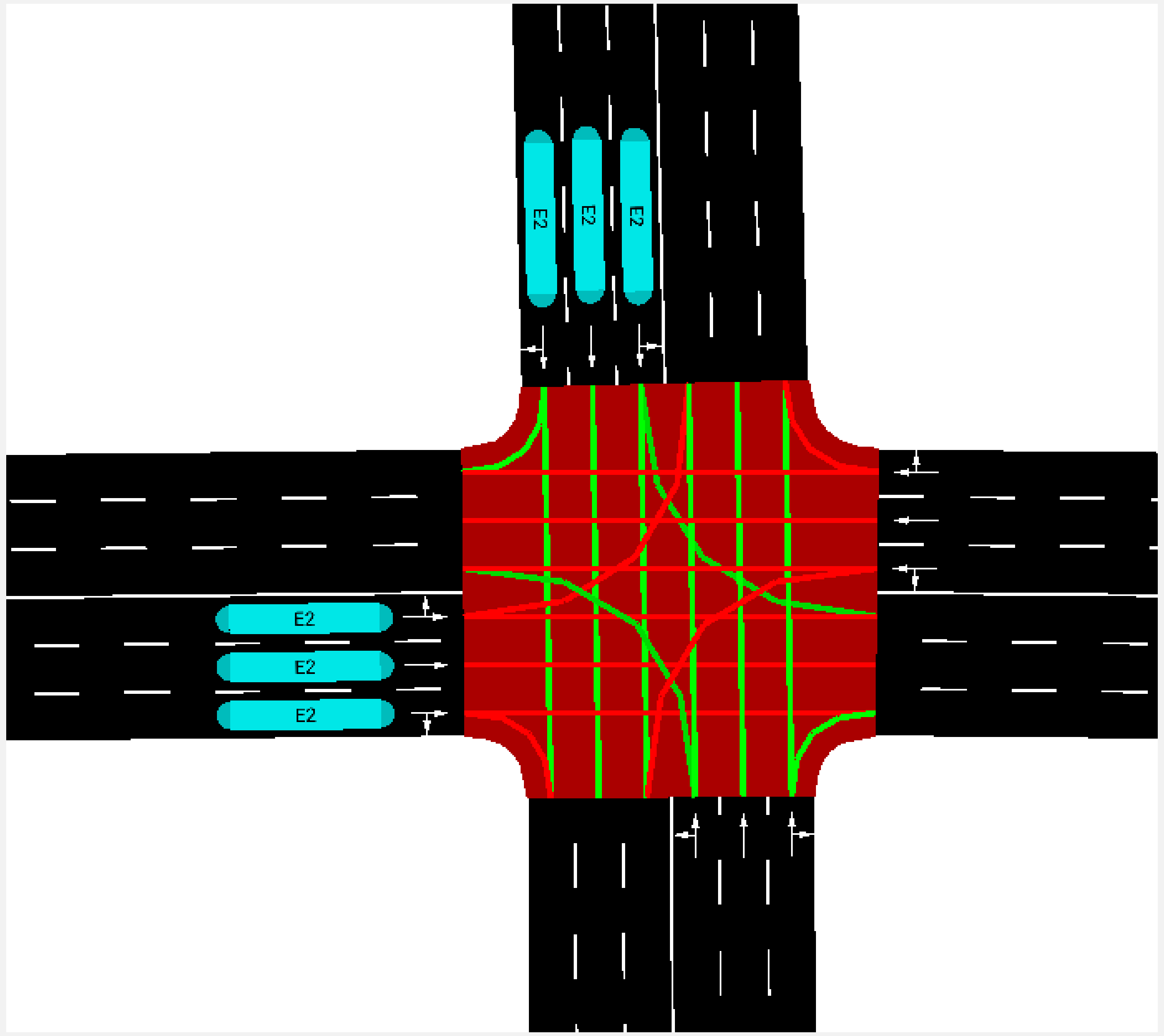
1. **Fixed Timing:**

2. **Q-Learning:**

3. **Deep Q-Learning:**

| Variable | ValueType | Description | Python Method |
|---|---|---|---|
| id list (0x00) | stringList | Returns a list of ids of all traffic lights within the scenario (the given Traffic Lights ID is ignored) | getIDList |
| count (0x01) | int | Returns the number of traffic lights within the scenario (the given Traffic Lights ID is ignored) | getIDCount |
| state (light/priority tuple) (0x20) | string | Returns the named tl's state as a tuple of light definitions from rRgGyYoO, for red, green, yellow, off, where lower case letters mean that the stream has to decelerate | getRedYellowGreen-State |
| default current phase duration (0x24) | double | Returns the default total duration of the currently active phase in seconds; To obtain the remaining duration use *(getNextSwitch() - simulation.getTime())*; to obtain the spent duration subtract the remaining from the total duration | getPhaseDuration |
| controlled lanes (0x26) | stringList | Returns the list of lanes which are controlled by the named traffic light. Returns at least one entry for every element of the phase state (signal index)[(1)(2)]. | getControlledLanes |
| controlled links (0x27) | compound object | Returns the links controlled by the traffic light, the index in the returned list corresponds to the tls link index of the connection. Each index maps to a list of link objects that share the same link index. Each link object is described by giving the incoming, outgoing, and via lane. | getControlledLinks |
| current phase (0x28) | int | Returns the index of the current phase in the current program | getPhase |
| current program (0x29) | string | Returns the id of the current program | getProgram |
| complete definition (light/priority tuple) (0x2b) | compound object | Returns the complete traffic light program, structure described under data types | getCompleteRedYel-lowGreenDefinition |
| assumed time of next switch (0x2d) | double | Returns the assumed time (in seconds) at which the tls changes the phase. Please note that the time to switch is not relative to current simulation step (the result returned by the query will be absolute time, counting from simulation start); to obtain relative time, one needs to subtract current simulation time from the result returned by this query. Please also note that the time may vary in the case of actuated/adaptive traffic lights | getNextSwitch |
| spent duration (0x38) | double | Returns the time spent in the current phase (in seconds) | getSpentDuration |

# E. Adding Traffic Light



# E. Adding Traffic Light

## 1. Fixed Timing:

Traci5.py

D: > OneDrive - York University > Bussiness2 > CodingPractical > RoadWayVRChannel > SUMO2Unity > SUMO > Tutorials > Detectors > traci5.FT.py > ...

```python
# Step 1: Add modules to provide access to specific libraries and functions
import os  # Module provides functions to handle file paths, directories, environment variables
import sys  # Module provides access to Python-specific system parameters and functions
import random
import numpy as np
import matplotlib.pyplot as plt  # Visualization






# Step 2: Establish path to SUMO (SUMO_HOME)
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("Please declare environment variable 'SUMO_HOME'")

# Step 3: Add Traci module to provide access to specific libraries and functions
import traci  # Static network information (such as reading and analyzing network files)

# Step 4: Define Sumo configuration
Sumo_config = [
    'sumo-gui',
    '-c', 'Detector.sumocfg',
    '--step-length', '0.10',
    '--delay', '1000',
    '--lateral-resolution', '0'
]

# Step 5: Open connection between SUMO and Traci
traci.start(Sumo_config)
traci.gui.setSchema("View #0", "real world")
```

## 2. Q-Learning:

Traci6.py

```python
# Step 1: Add modules to provide access to specific libraries and functions
import os  # Module provides functions to handle file paths, directories, environment variables
import sys  # Module provides access to Python-specific system parameters and functions
import random
import numpy as np
import matplotlib.pyplot as plt  # Visualization






# Step 2: Establish path to SUMO (SUMO_HOME)
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("Please declare environment variable 'SUMO_HOME'")

# Step 3: Add Traci module to provide access to specific libraries and functions
import traci  # Static network information (such as reading and analyzing network files)

# Step 4: Define Sumo configuration
Sumo_config = [
    'sumo-gui',
    '-c', 'Detector.sumocfg',
    '--step-length', '0.10',
    '--delay', '1000',
    '--lateral-resolution', '0'
]

# Step 5: Open connection between SUMO and Traci
traci.start(Sumo_config)
traci.gui.setSchema("View #0", "real world")

# -------------------------
# Step 6: Define Variables
# -------------------------
```

## 3. Deep Q-Learning:

Traci7.py

```python
# Step 1: Add modules to provide access to specific libraries and functions
import os  # Module provides functions to handle file paths, directories, environment variables
import sys  # Module provides access to Python-specific system parameters and functions
import random
import numpy as np
import matplotlib.pyplot as plt  # Visualization

# Step 1.1: (Additional) Imports for Deep Q-Learning
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Step 2: Establish path to SUMO (SUMO_HOME)
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("Please declare environment variable 'SUMO_HOME'")

# Step 3: Add Traci module to provide access to specific libraries and functions
import traci  # Static network information (such as reading and analyzing network files)

# Step 4: Define Sumo configuration
Sumo_config = [
    'sumo',
    '-c', 'Detector.sumocfg',
    '--step-length', '0.10',
    '--delay', '1000',
    '--lateral-resolution', '0'
]

# Step 5: Open connection between SUMO and Traci
traci.start(Sumo_config)
#traci.gui.setSchema("View #0", "real world")
```

**In the Terminal:**

**Write pip list and check numpy, matplotlib, tensorflow; if you don't have, then**

- ❑ **pip install numpy**
- ❑ **pip install matplotlib**
- ❑ **Pip install tensorflow**

# E. Adding Traffic Light (Q-Learning)

$$\max_{\{a(t)\}} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^{t}\, r(t)\right]$$

**1. Max Rewards**

$$r(t) = -\left(\sum_{i=0}^{2} q_{EB,i}(t) + \sum_{j=0}^{2} q_{SB,j}(t)\right)$$

2. Rewards (-Queue Length)

$$s(t) = \big(q_{EB,0}(t),\ q_{EB,1}(t),\ q_{EB,2}(t),\ q_{SB,0}(t),\ q_{SB,1}(t),\ q_{SB,2}(t),\ \mathrm{phase}(t)\big)$$

3. State

$$q_{\ell}(t+1) = \max\Big\{0,\ q_{\ell}(t) + \mathrm{arrivals}_{\ell}(t) - \mathrm{departures}_{\ell}\big(t, \mathrm{phase}(t)\big)\Big\}$$

4. Each lane's queue at time $t$+1 depends on the old queue, plus arrivals, minus departures

$$\mathrm{phase}(t+1) = \begin{cases} \mathrm{phase}(t), & \text{if } a(t) = 0, \\ \big(\mathrm{phase}(t) + 1\big) \bmod N, & \text{if } a(t) = 1 \text{ and } t - t_{\text{last switch}} \geq G_{\min}. \end{cases}$$

5. Action (Switch Phase and Min Green Time)

$$Q(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \alpha\left[r(t) + \gamma \max_{a'} Q(s(t+1), a') - Q(s(t), a(t))\right]$$

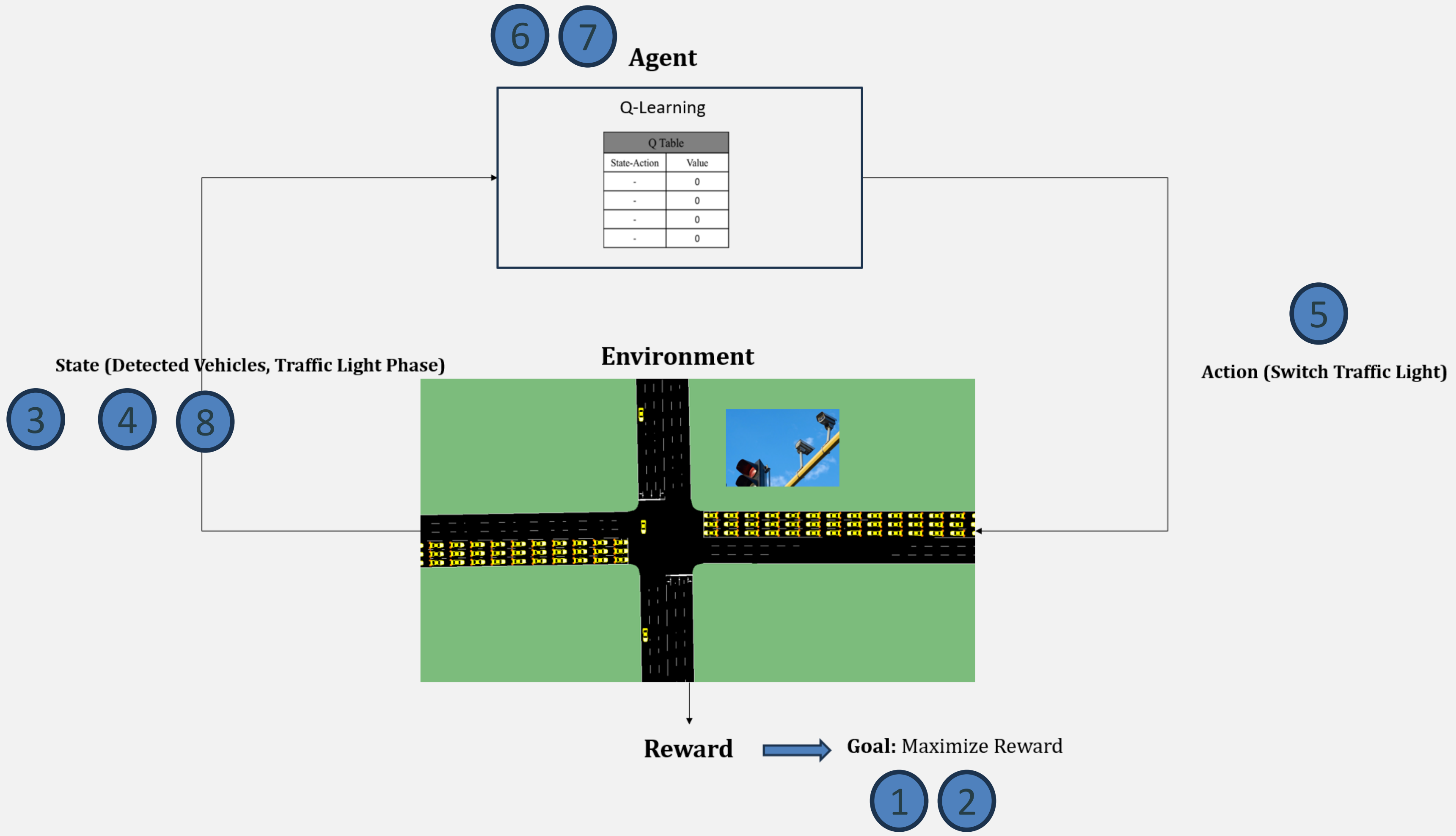
6. Q-Learning Update

$$\pi_{\varepsilon}(a \mid s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|}, & \text{if } a = \arg\max_{a'} Q(s, a'), \\ \frac{\varepsilon}{|\mathcal{A}|}, & \text{otherwise.} \end{cases}$$

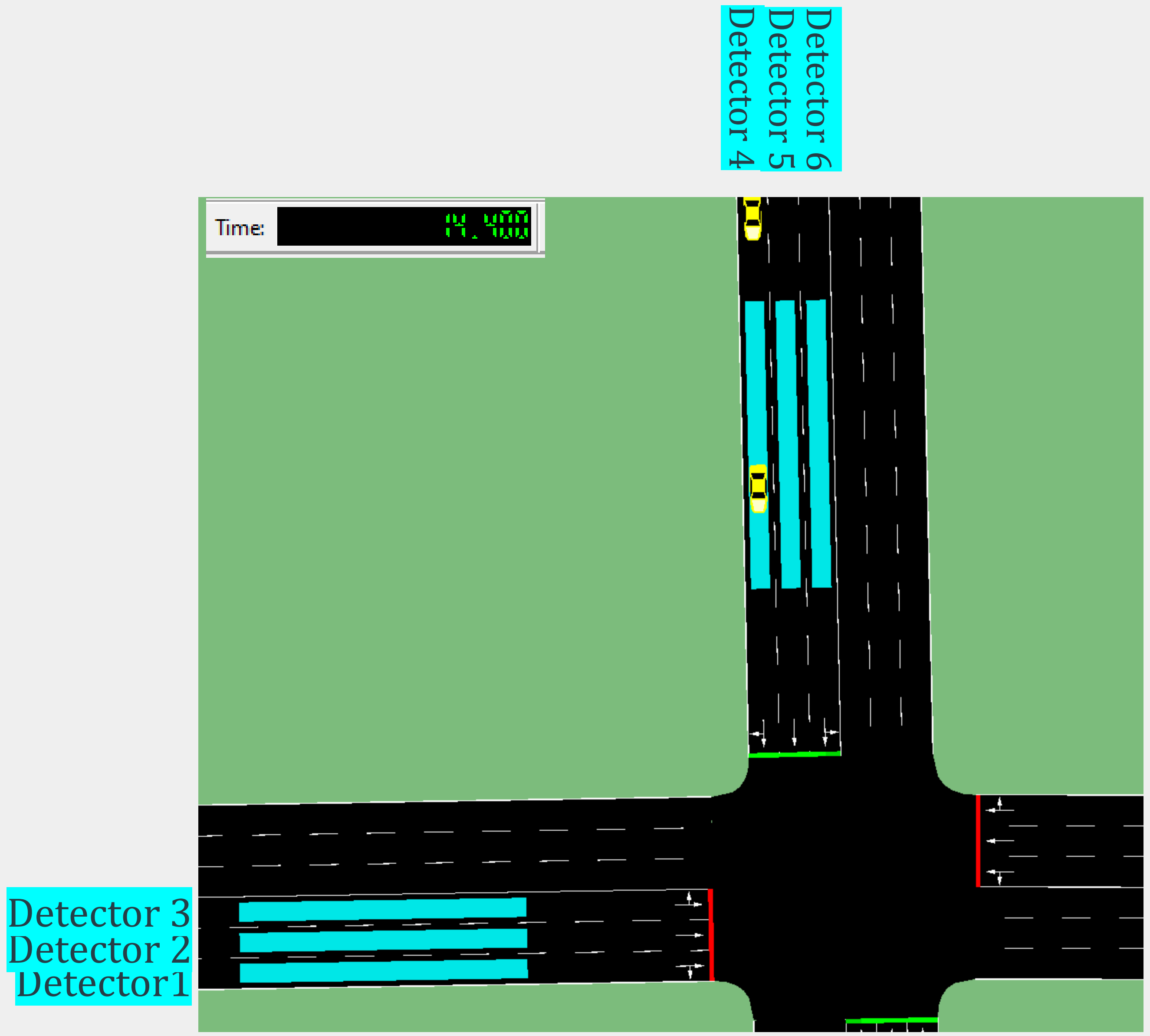
7. the exploration (Randomly or Greedy)

$$q_{\ell}(t) \geq 0, \quad \mathrm{phase}(t) \in \{0, 1, \ldots, \mathrm{numPhases} - 1\}.$$

8. Variables get From SUMO



# Q-learning


Detector 6
Detector 5
Detector 4
Time:
Detector 3
Detector 2
Detector1

# Q-learning

- Assume we are in step 630, we have below result:

```
Step 630, Current_State: (2, 2, 0, 1, 1, 0, 0), Action: 1, New_State: (2, 2, 0, 1, 1, 0, 1), Reward: -6.00, Cumulative Reward: -1489.00,
```

- The Q-table until step 630 (0-629):

```
(2, 2, 0, 1, 0, 0, 2) -> [-0.95 -0.5 ]
(2, 2, 0, 1, 0, 0, 3) -> [-0.5   -0.545]
(2, 2, 0, 0, 1, 0, 2) -> [-0.5 -0.6]
(2, 2, 0, 1, 1, 0, 3) -> [-0.6 -0.6]
(2, 2, 0, 1, 1, 0, 0) -> [-0.6  0. ]
```

- How to calculate Q-Value and Update Q-Table for step 630?

1. **Old (Current) State:**

$$s = (2,2,0,1,1,0,0)$$

2. **Action:**

$$a = 1$$

3. **Reward:**

$$r = -6.0$$

4. **New State:**

$$s' = (2,2,0,1,1,0,1)$$

5. **Old Q-value** now for **Action 1:**

$$Q_{\text{old}} = Q((2,2,0,1,1,0,0),1)=0$$

6. **Max future Q-value:**

$$\max_{a'} Q((2,2,0,1,1,0,1),a').$$

Again, apply the update:

$$\begin{aligned} Q_{\text{new}} &= 0.0 \;+\; 0.1\Big[-6.0 + 0.9 \times 0.0 - 0.0\Big] \\ &= 0.1 \times (-6.0) \\ &= -0.6. \end{aligned}$$

the Q-values for `(2, 2, 0, 1, 1, 0, 0)` become:

```css
[-0.6,  -0.6]
```

## 1. Q-learning Update Formula Recap

Your code uses the **standard Q-learning** update:

$$Q(s,a) \;\leftarrow\; Q(s,a) \;+\; \alpha\Big[r \;+\; \gamma \max_{a'} Q(s',a') \;-\; Q(s,a)\Big].$$

Where:

- $s$ = current (old) state
- $a$ = action taken
- $s'$ = next (new) state
- $r$ = immediate reward
- $\alpha$ = learning rate (0.1 in your code)
- $\gamma$ = discount factor (0.9 in your code)
- $\max_{a'} Q(s',a')$ = maximum Q-value for the **new state** over all possible actions $a'$.

# E. Traffic Light (Q-Learning)

**Output 1:** State, Action, Reward

#Cars in Detector 1
#Cars in Detector 3
#Cars in Detector 6
Traffic Phase

```
Step 89, State: (0, 0, 0, 1, 0, 0, 0), Action: 1,  Reward: -1.00, Cumulative Reward: -5.00
```

#Cars in Detector 5
#Cars in Detector 4
#Cars in Detector 2

**Output 2:** Q-Table & Q Values

```
Online Training completed. Final Q-table size: 2440
State: (0, 0, 0, 0, 0, 0, 0) -> Q-values: [0. 0.]
State: (0, 0, 0, 0, 0, 0, 1) -> Q-values: [0. 0.]
State: (0, 0, 0, 0, 0, 0, 2) -> Q-values: [-0.1  0. ]
State: (0, 0, 0, 1, 0, 0, 2) -> Q-values: [-0.29701 -0.45929]
State: (0, 0, 0, 1, 0, 0, 3) -> Q-values: [-0.29701 -0.29701]
State: (0, 0, 0, 1, 0, 0, 0) -> Q-values: [-0.29701 -0.29701]
```

Action 0 Action 1

```
Q-values: [0. 0.]
```

Reward if Action 0 Reward if Action 1

```
[-0.29701 -0.29701]
```

```
State: (0, 0, 0, 2, 0, 0, 2) -> Q-values: [-1.00927182 -0.642134  ]
```

???

# E. Adding Traffic Light (Deep Q-Learning)

$$Q(s(t),a(t)) \leftarrow Q(s(t),a(t)) + \alpha\left[r(t) + \gamma \max_{a'} Q(s(t+1),a') - Q(s(t),a(t))\right]$$

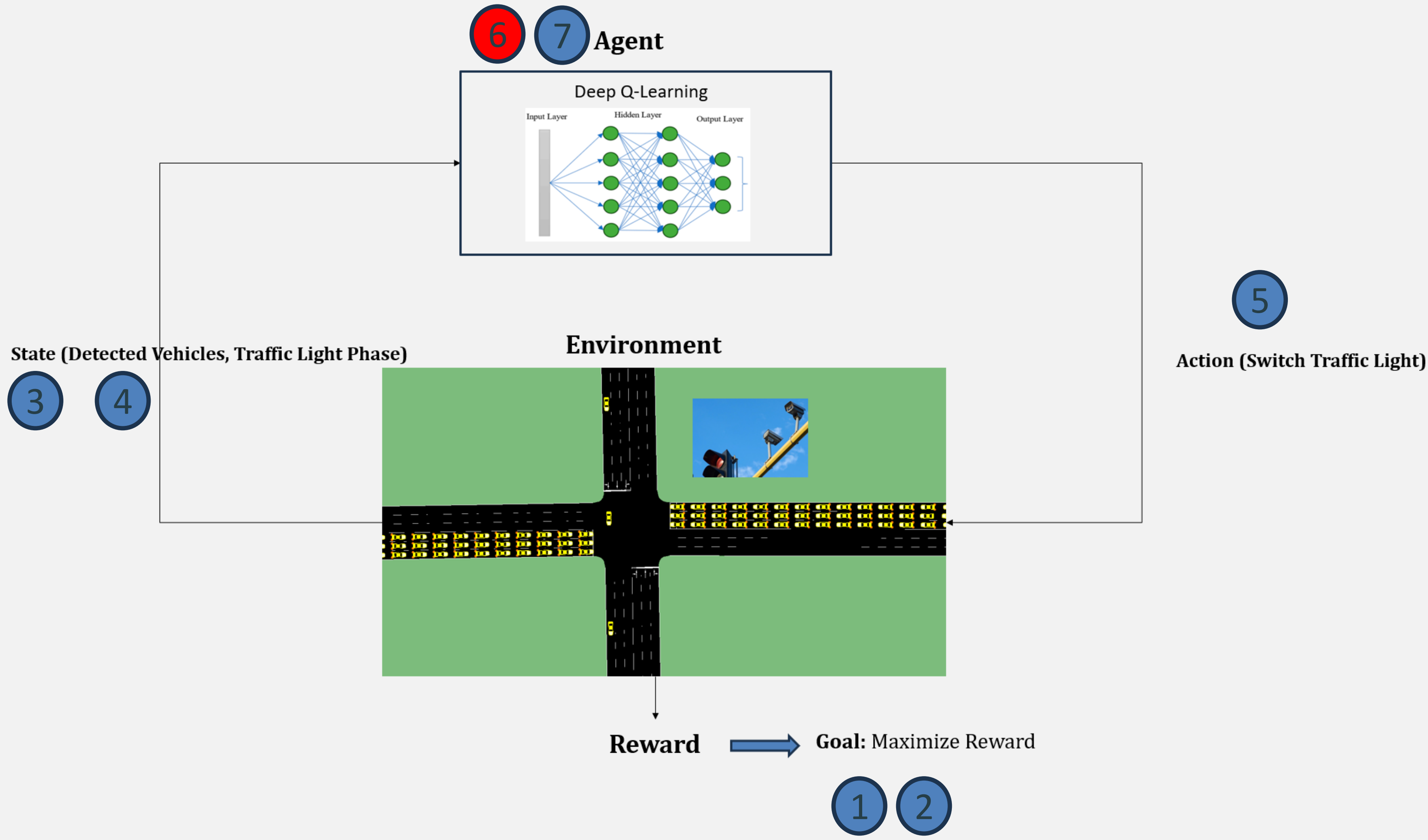
6. In Q-Learning

Change to this

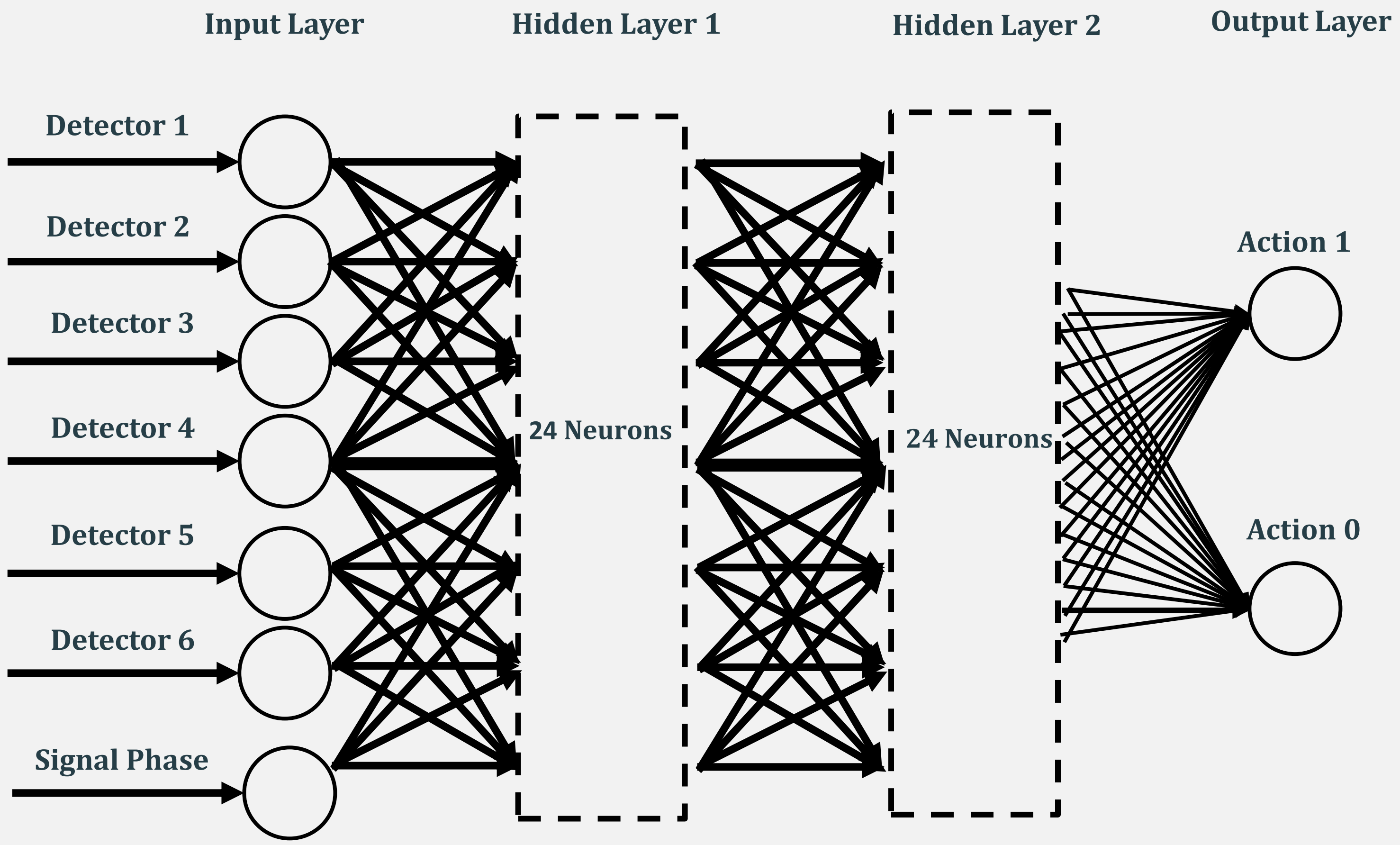
$$Q_\theta(s(t),a(t)) \;\leftarrow\; Q_\theta(s(t),a(t)) \;+\; \text{optimizer step}\left[\left(y - Q_\theta(s(t),a(t))\right)^2\right].$$

1. **Forward Pass:**
   Evaluate the current Q-network to get predicted Q-values:
   $$Q_\theta(s) = \mathrm{NN}(s;\theta).$$
   For each state $s$, it outputs a vector (Q-value for each action $a \in \{0,1\}$).
2. **Target Calculation:**
   For the sampled transition $\left(s(t),a(t),r(t),s(t+1)\right)$, compute
   $$y \;=\; r(t) + \gamma \max_{a'} Q_\theta\left(s(t+1),a'\right).$$
3. **Network Update (Stochastic Gradient Descent):**
   Adjust $\theta$ to reduce the **Mean Squared Error**:
   $$L(\theta) \;=\; \left[y \;-\; Q_\theta(s(t),a(t))\right]^2.$$

6. In Deep Q-Learning



# E. Traffic Light (Deep Q-Learning)




**Activation Layer:** Rectified Linear Unit
**Loss Function:** Mean squared error (MSE)
**Optimizer:** Adam (learning_rate=0.001)

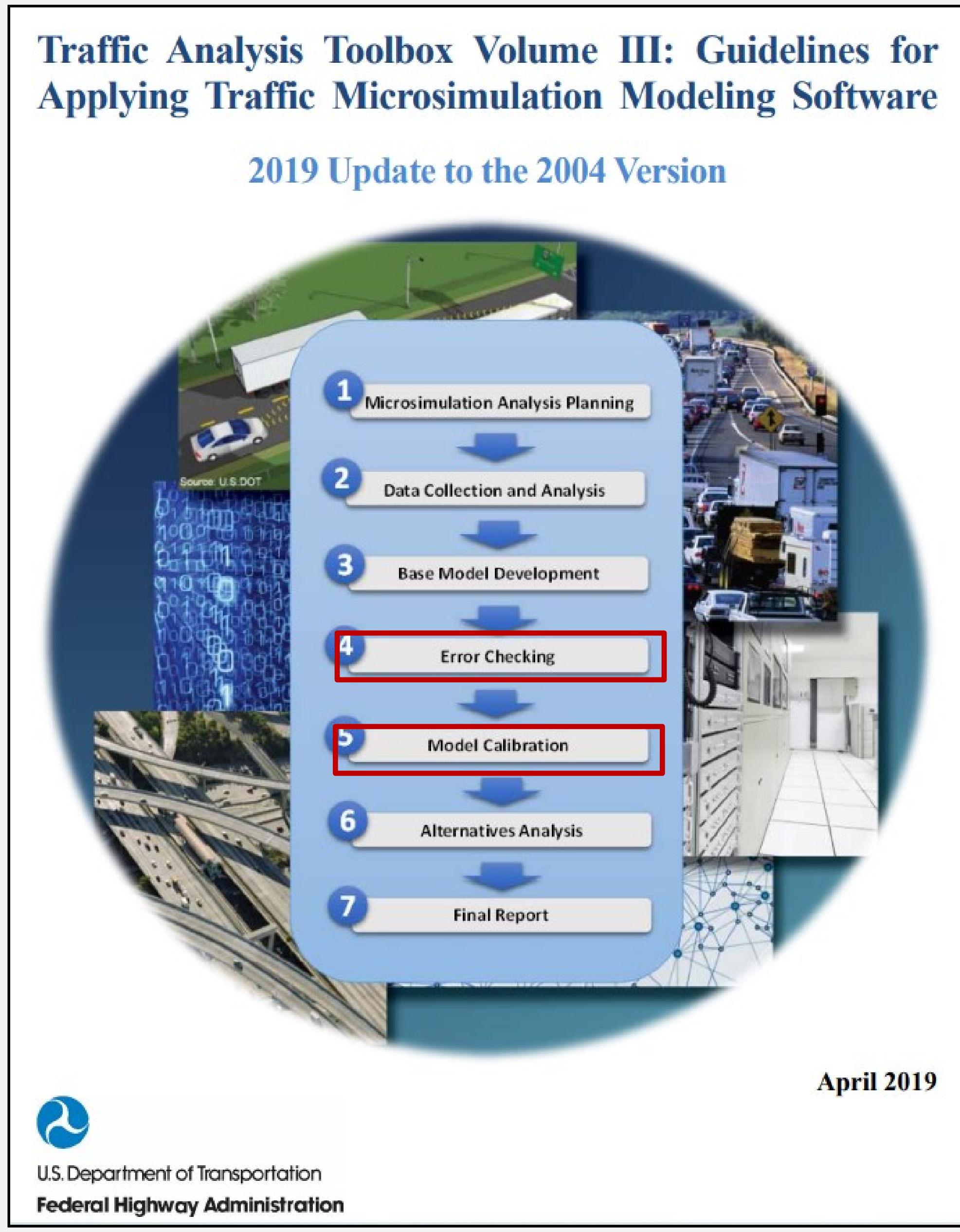
**Reference:** Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

# Step 4&5: Error Checking and Model Calibration

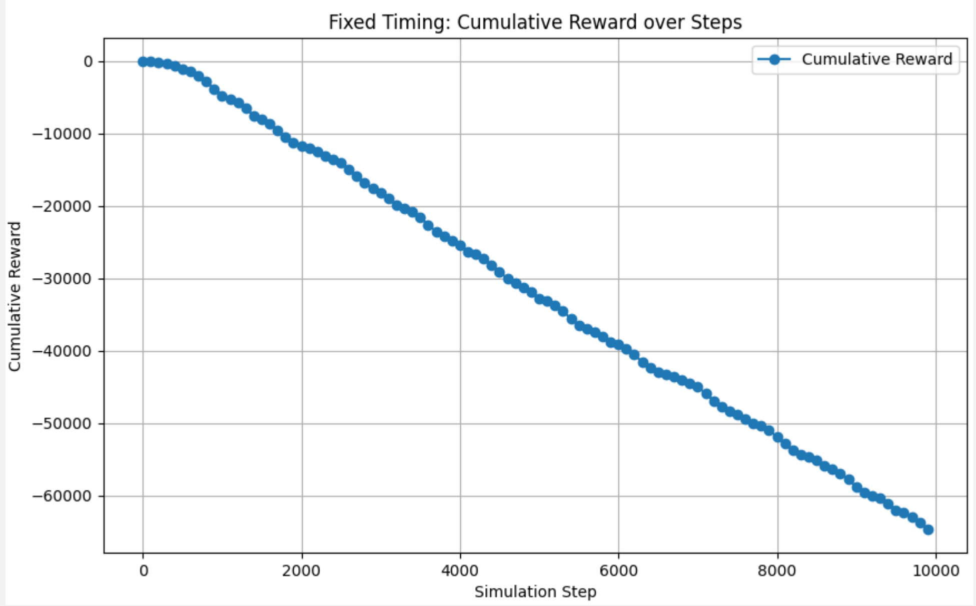
**We skip these steps:**

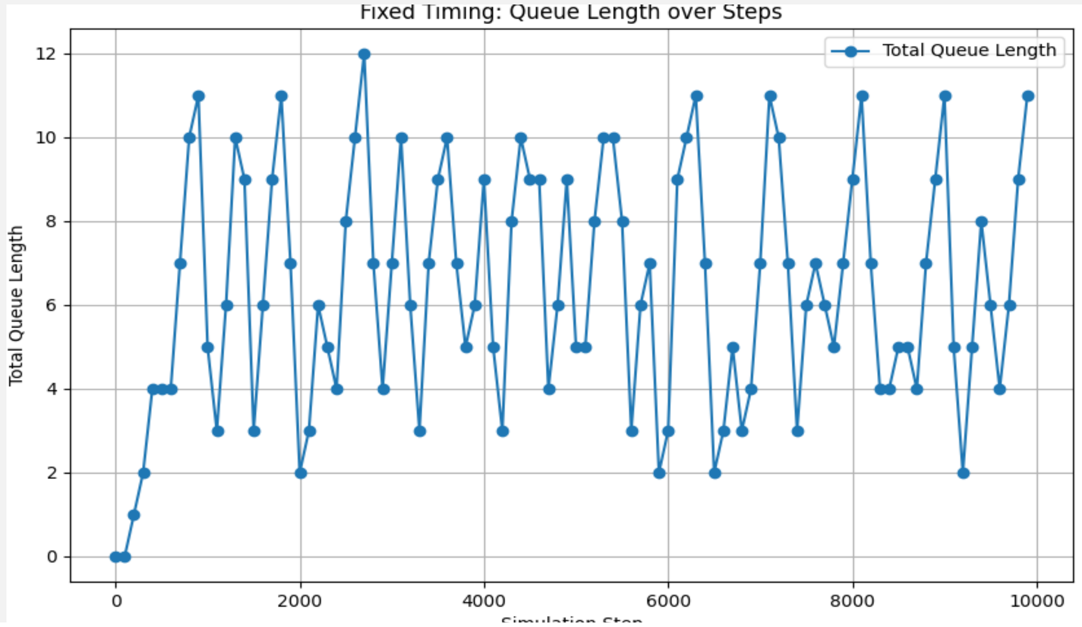
**Step 4: Error Checking:** Is usually checking any bottleneck or geometry failure

**Step 5: Model Calibration:** Calibrate the model with real-world data

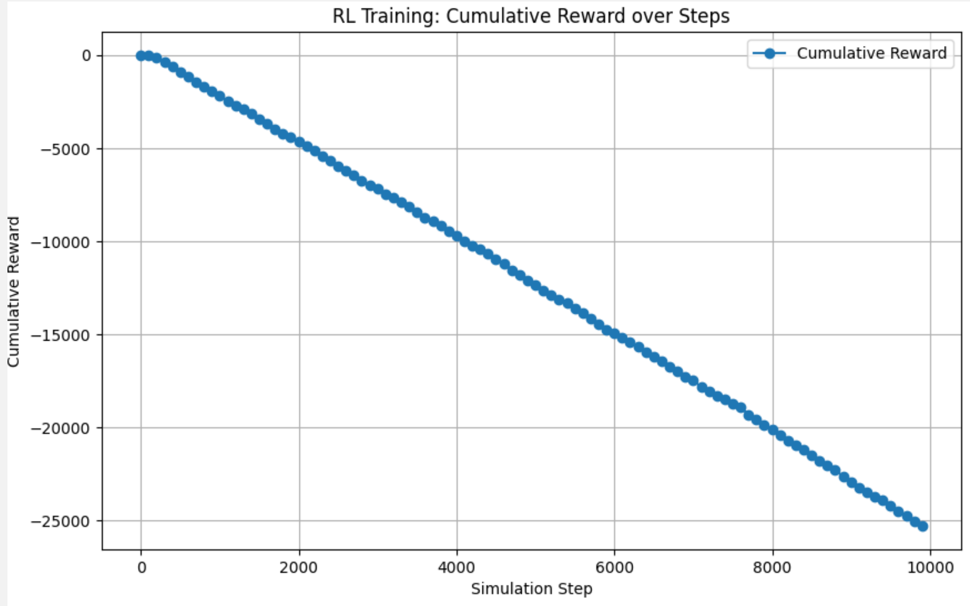


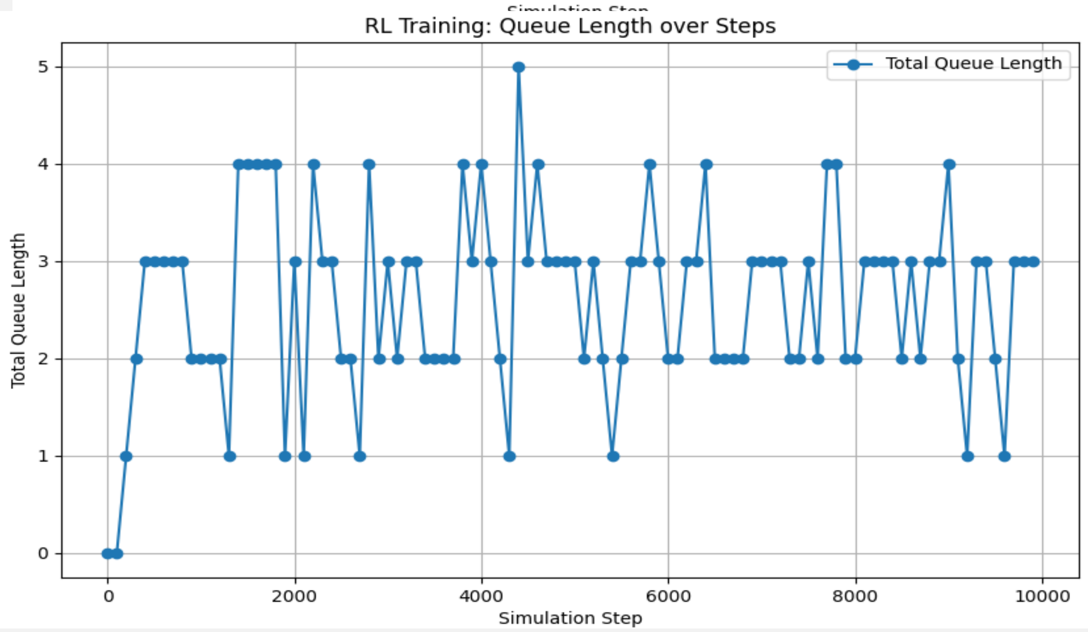
# Step 6: Alternative Analysis



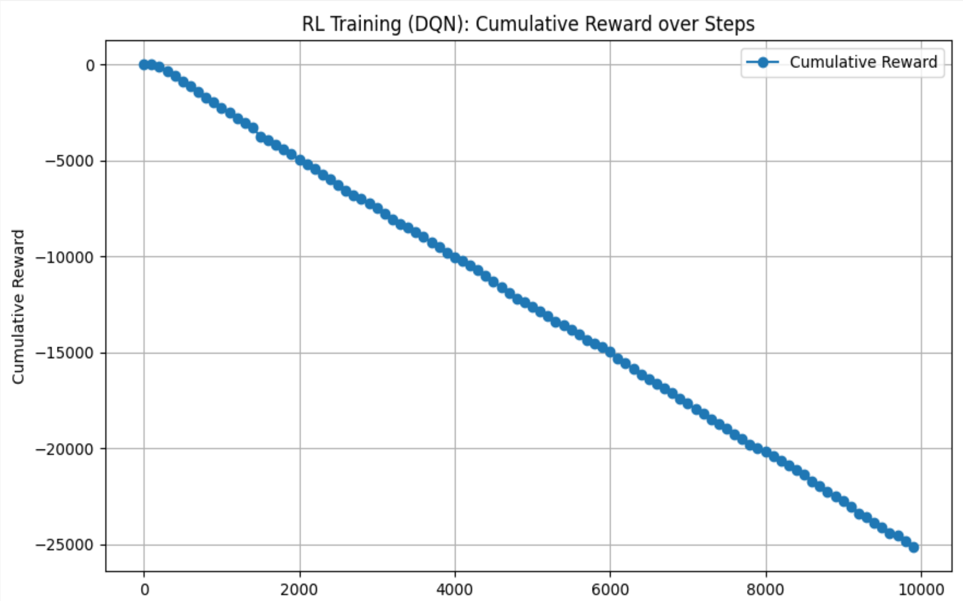



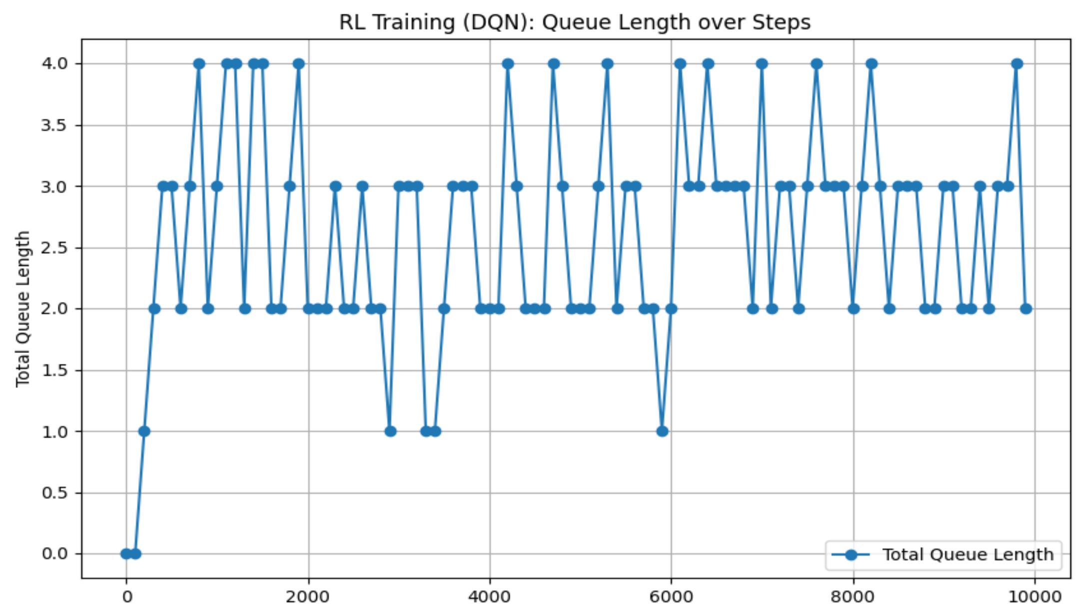

Fixed Timing



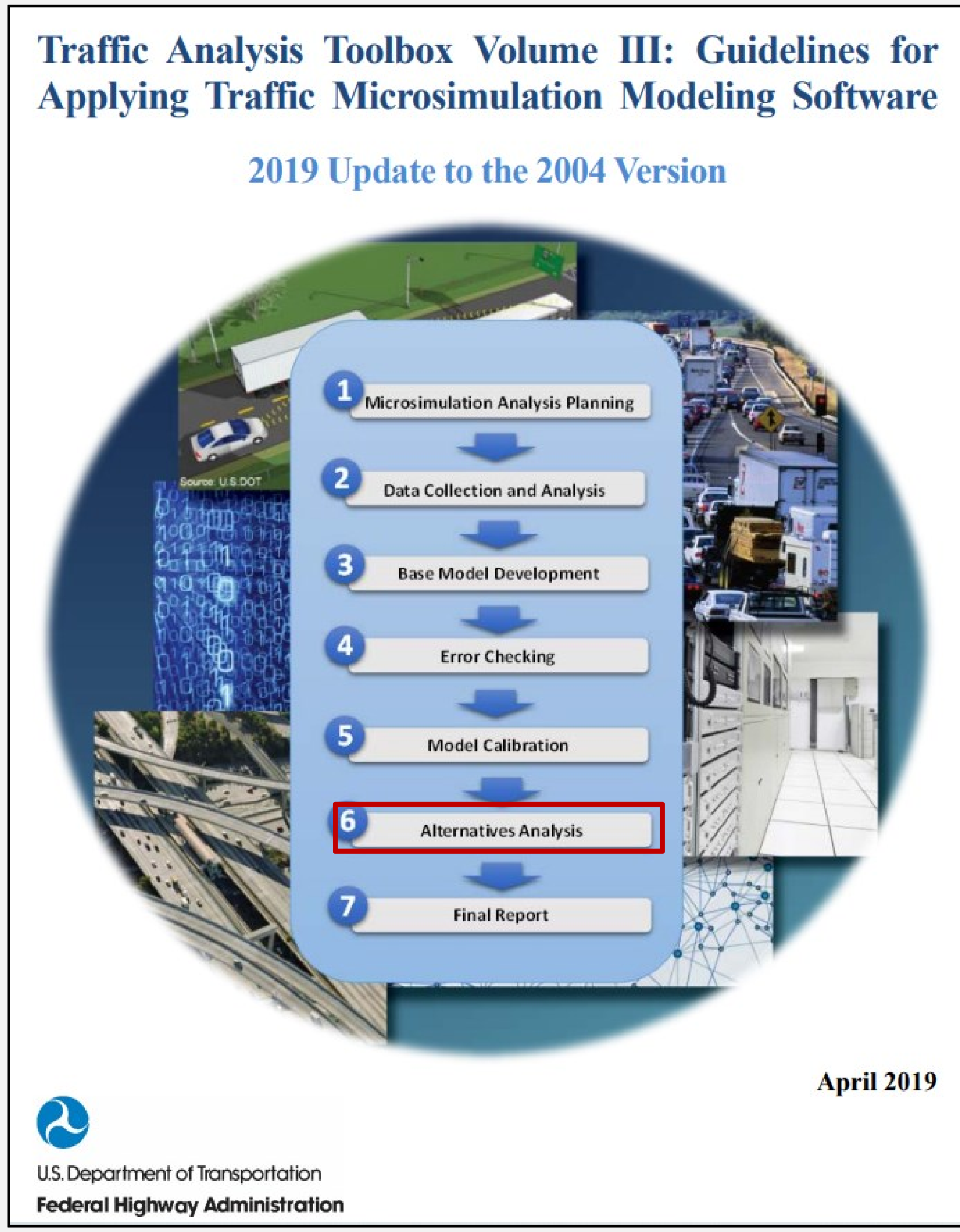




Q-Learning




Deep Q-Learning


Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software

2019 Update to the 2004 Version



April 2019

U.S. Department of Transportation
Federal Highway Administration