Optimizing Multi-Intersection Traffic Signal Control Using Deep Reinforcement Learning

Abdul Basit & M. Abdullah

LUMS

December 18, 2025

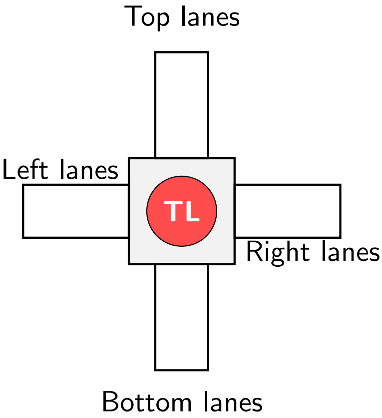
Traffic Signal Control as an MDP Problem

- **Problem Statement:** Optimize traffic signal timing to minimize congestion and maximize throughput
- **MDP Components:**
  - States (S): Traffic conditions at intersections
  - Actions (A): Signal phase selections
  - Transitions (P): Traffic flow dynamics (SUMO simulator)
  - Rewards (R): Negative congestion + positive throughput
- **Objective:** Find policy π^* that maximizes expected cumulative reward

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi\right]$$

Single Intersection MDP Setup

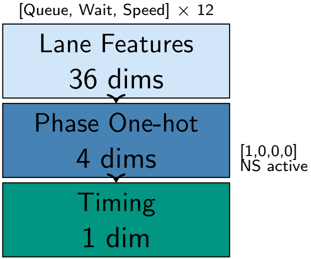
- **Environment:** One traffic light controlling 4 approaches (N, E, S, W)
- **Simulation:** SUMO microscopic traffic simulator with TraCI interface
- **Decision Frequency:** Every 5 seconds (action execution time)
- **Episode Length:** 3600 seconds (1 hour simulation)
- **Traffic Light Program:** 8 phases (4 green + transitions)
- **Agent Control:** Selects from green phases (0, 2, 4, 6)
- **Safe Transitions:** Environment handles yellow → red → green automatically



SUMO Phase Cycle: 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 0

Single Intersection State Space (41 Dimensions)

- **State Vector Structure:** [lane_features, phase_encoding, timing]
- **Lane Features (36 dimensions):**
  - 12 incoming lanes × 3 metrics each
  - **Queue Length:** Number of halted vehicles per lane
  - **Waiting Time:** Sum of waiting times for vehicles in lane
  - **Speed:** Average normalized speed (0-1) / lane
- **Phase Encoding (4 dimensions):**
  - One-hot vector for current green phase
  - [1,0,0,0] = Phase 0 (NS through/right)
  - [0,1,0,0] = Phase 2 (NS left)
  - [0,0,1,0] = Phase 4 (EW through/right)
  - [0,0,0,1] = Phase 6 (EW left)
- **Timing Feature (1 dimension):**
  - Time since last phase change (normalized 0-1)

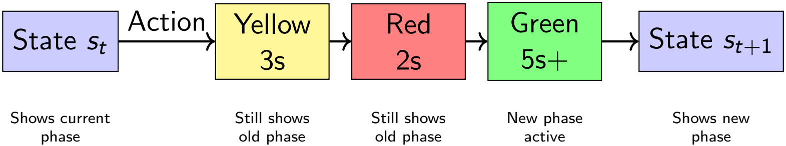


Total: $36 + 4 + 1 = 41$ dimensions

Single Intersection State Space - Properties

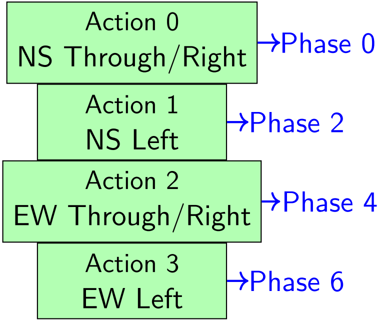
- **State Update Timing:**
  - State reflects conditions **before** action execution
  - During transitions: shows **previous** green phase
  - Updates only after reaching new green phase

- **Why This State Representation?**
  - Captures local traffic congestion patterns
  - Phase information enables temporal learning
  - Normalized features handle varying traffic volumes
  - Queue lengths indicate immediate congestion
  - Waiting times measure cumulative delay
  - Speed indicates flow quality



Single Intersection Action Space (4 Actions)

- **Action Definition:** Direct selection of green traffic phases. Each state corresponds to SUMO phase with 12-character string
- **Action 0:** Phase 0 - NS through/right green (SUMO phase 0)
- **Action 1:** Phase 2 - NS left green (SUMO phase 2)
- **Action 2:** Phase 4 - EW through/right green (SUMO phase 4)
- **Action 3:** Phase 6 - EW left green (SUMO phase 6)



Single Intersection Action Execution

- **Action Execution Process:**
  1. Agent selects action (0, 1, 2, or 3)
  2. Environment checks: action $\neq$ current_phase?
  3. **If different phase:**
     - Set yellow phase for 3 seconds
     - Set red clearance phase for 2 seconds
     - Set new green phase
     - Update current_phase and reset timer
  4. **If same phase:** Continue current green phase
  5. Execute simulation for 5 seconds with active phase
  6. Compute reward and update state

- **Safe Transition Details:**
  - Yellow phase = current_phase + 1 (always)
  - Red clearance uses all-red pattern phase
  - New green = green_phases[action]
  - All transitions use SUMO's built-in phase program

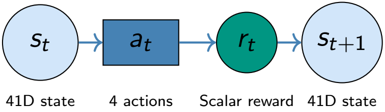
Single Intersection MDP Components

- **States (S):** 41-dimensional continuous vectors
  - Lane features + phase encoding + timing
- **Actions (A):** 4 discrete phase selections {0,1,2,3}
- **Transitions (P):** Deterministic given SUMO physics
  - $s_{t+1} = SUMO\_dynamics(s_t, a_t, traffic\_arrivals)$
- **Rewards (R):** Traffic efficiency metric
  - Penalizes congestion, rewards throughput
- **Discount Factor (γ):** 0.95
- **Horizon:** Finite episodes (3600s)

Reward Function

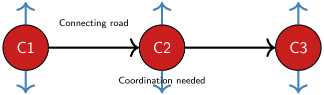
$$R = -0.5 \cdot W_t - Q_t + 500 \cdot T_t - P_t$$

- W_t = Total waiting time
- Q_t = Total queue length
- T_t = Throughput
- P_t = Congestion penalty



Multi-Intersection MDP Setup (3 Intersections)

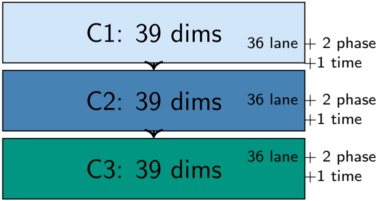
- **Environment:** 3 connected intersections (C1, C2, C3) in network
- **Coordination Challenge:** Joint control of correlated traffic flows
- **Network Topology:** Linear arrangement with connecting roads
- **Simulation:** Same SUMO setup with multi-TraCI connections
- **Decision Frequency:** Every 5 seconds (synchronized across intersections)
- **Episode Length:** 3600 seconds
- **Traffic Lights:** Each with 4-phase program (2 controllable greens)
- **Single Agent:** Controls entire network for coordinated optimization



Each intersection:
12 lanes, 4 phases,
2 controllable greens

Multi-Intersection State Space (117 Dimensions)

- **State Structure:** Concatenated per-intersection features
- **Formula:** State = [C1_features, C2_features, C3_features]
- **Per-Intersection Features (39 dimensions each):**
  - **Lane Features (36 dims):** 12 lanes $\times$ 3 metrics
    - Queue length, waiting time, normalized speed per lane
  - **Phase Encoding (2 dims):** One-hot for current green phase
    - [1,0] = NS green (Phase 0), [0,1] = EW green (Phase 2)
  - **Timing (1 dim):** Time since last phase change

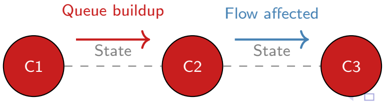


Total: 117 dimensions

- **Total Dimensions:** 3 intersections $\times$ 39 features = 117

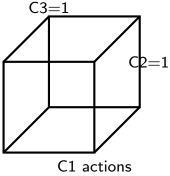
Multi-Intersection State Space - Properties

- **Interdependence:** States capture cross-intersection traffic patterns
  - Queue buildup at C1 affects C2's incoming traffic
  - Phase coordination enables green wave effects
- **Phase Coordination:** Agent sees all intersection phases simultaneously
  - Can learn to synchronize compatible phases
  - Prevents conflicting traffic flows between intersections
- **Queue Propagation:** Downstream queues affect upstream decisions
  - Agent learns network-wide congestion dynamics
  - Enables proactive traffic management
- **Update Timing:** All intersections update states simultaneously
  - Synchronized across the entire network
  - Consistent state representation for coordinated actions



Multi-Intersection Action Space (8 Joint Actions)

- **Individual Actions:** Each intersection has 2 actions
  - Action 0: NS green, Action 1: EW green
- **Joint Actions:** Cartesian product of individual actions
- **Action Space Size:** $2^3 = 8$ joint actions
- **Encoding Scheme:**
  - Joint action = C1_action + 2×C2_action + 4×C3_action
  - Example: [1,0,1] → 1 + 0 + 4 = Joint Action 5
- **Decoding:** Extract per-intersection actions from joint index

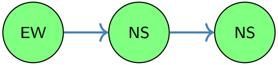


$2 \times 2 \times 2 = 8$ joint actions

Multi-Intersection Action Space - Interpretations

- **Joint Action Meanings:**
  1. **Action 0:** [NS, NS, NS] - All intersections in north-south green
  2. **Action 1:** [EW, NS, NS] - C1 east-west, others north-south
  3. **Action 2:** [NS, EW, NS] - C2 east-west, others north-south
  4. **Action 3:** [EW, EW, NS] - C1 & C2 east-west, C3 north-south
  5. **Action 4:** [NS, NS, EW] - C3 east-west, others north-south
  6. **Action 5:** [EW, NS, EW] - C1 & C3 east-west, C2 north-south
  7. **Action 6:** [NS, EW, EW] - C2 & C3 east-west, C1 north-south
  8. **Action 7:** [EW, EW, EW] - All intersections in east-west green

- **Execution Details:**
  - All intersections transition simultaneously
  - Each follows: green → yellow(3s) → red(2s) → new_green
  - Independent transitions but coordinated timing



Joint Action 1: [EW, NS, NS]

C1: EW green C2,C3: NS green

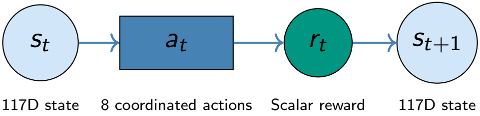
Multi-Intersection Network MDP

- **States (S):** 117-dimensional continuous vectors
- **Actions (A):** 8 discrete joint phase selections $\{0,1,2,3,4,5,6,7\}$
- **Transitions (P):** Network traffic dynamics
  - $s_{t+1} = SUMO\_network\_dynamics(s_t, a_t, traffic\_patterns)$
- **Rewards (R):** Network-wide efficiency
  - Penalizes congestion, rewards throughput
- **Discount Factor (γ):** 0.95
- **Horizon:** 3600s episodes

Network Reward Function

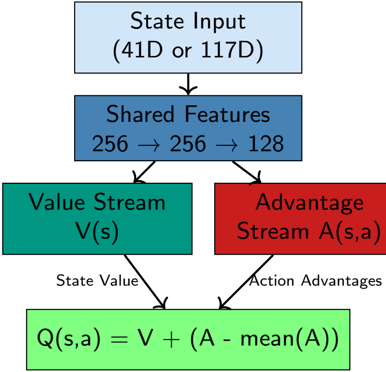
$$R = -0.5 \cdot W_t - Q_t + 100 \cdot T_t - P_t$$

- W_t = total waiting time in the network
- Q_t = total queue length
- T_t = network throughput



Dueling Double Deep Q-Network Architecture

- **Core Algorithm:** Dueling Double DQN combines two key innovations
- **Dueling Architecture:** Separates value and advantage estimation
- **Double DQN:** Reduces overestimation bias in Q-value estimation
- **Why for Traffic Control?**
  - High-dimensional state spaces (41D/117D)
  - Complex action coordination (4/8 actions)
  - Need for sample-efficient learning
  - Stability requirements for real-time control
- **Implementation:** Both enabled by default in my codebase



Dueling DQN Architecture

- **Traditional DQN:** Directly predicts $Q(s, a)$
  - No separation of state value vs. action advantage
  - Inefficient learning of state values
- **Dueling DQN:** Separates value and advantage
  - **Value Stream:** $V(s)$ – value of being in state s
  - **Advantage Stream:** $A(s, a)$ – advantage of action a in state s
  - **Combination:** $Q(s, a) = V(s) + (A(s, a) - \bar{A}(s))$

Mathematical Formulation

$$Q(s, a) = V(s) + [A(s, a) - \bar{A}(s)]$$

$$\text{where: } \bar{A}(s) = \frac{1}{|A|}\sum_{a'} A(s, a')$$

- **Why subtract the mean?**
  - Ensures advantage has zero mean (identifiability)
  - Makes $V(s)$ the true state value
  - Prevents gradient vanishing

Advantages of Dueling DQN Architecture

- **Sample Efficiency:** Value stream learns state values; advantage stream learns action preferences. Fewer samples needed.
- **Better Generalization:** State values learned independently of actions. Can handle unseen action combinations, useful for traffic control.
- **Stable Learning:** Separates state and action learning, reducing interference and improving gradient flow.
- **Interpretability:** Analyze learned state values and action advantages for understanding policies.

Why Perfect for Traffic Control?

State values matter regardless of actions; action advantages depend on traffic patterns. Efficient learning and coordination across intersections.

Understanding Double DQN Algorithm

- **Problem: Overestimation Bias in DQN**
  - Same network selects and evaluates actions
  - Inflated Q-values $\rightarrow$ suboptimal policies
- **Double DQN Solution**
  - **Policy Network:** selects best action
  - **Target Network:** evaluates that action
  - Decouples selection and evaluation

Comparison

Standard DQN:

$$Q_{\theta^-}(s', \arg\max_{a'} Q_{\theta^-}(s', a'))$$

Double DQN:

$$Q_{\theta^-}(s', \arg\max_{a'} Q_{\theta}(s', a'))$$

- **Implementation**
  - Online network: $a' = \arg\max_{a'} Q_\theta(s', a')$
  - Target network: $Q_{\theta^-}(s', a')$

Advantages of Double DQN Algorithm

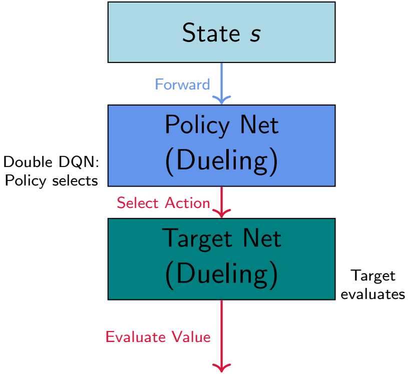
- **Reduces Overestimation Bias:**
  - More accurate Q-values $\rightarrow$ better policies
  - Important in sparse reward environments
- **More Stable Training:**
  - Less local optima, better convergence
  - Reduces noise impact
- **Better Exploration:**
  - Accurate Q-values $\rightarrow$ smarter exploration
  - ϵ-greedy more effective
- **Improved Performance:**
  - Outperforms standard DQN
  - Crucial for complex tasks like traffic control

Why Critical for Traffic Control?

- Accurate Q-values for safety-critical decisions
- Overestimation could lead to dangerous traffic patterns
- Ensures stability for real-time control

Dueling Double DQN: Best of Both Worlds

- **Combined Architecture:**
  - Dueling for efficient value learning
  - Double for accurate value estimation
  - Target networks for stability
  - Experience replay for sample efficiency
- **Training Process:**
  1. Sample batch from experience replay
  2. Use policy network to select actions (Double DQN)
  3. Use target network to evaluate values (Double DQN)
  4. Update dueling network parameters
  5. Periodically update target network



Q-target $= r + \gamma \cdot$ Q-target(s', a')

Implementation Specifications

Network Topology (Dueling)

- **Input Layer:**
  - 41D (Single) or 117D (Multi)
- **Shared Feature Extractor:**
  - FC Layers: $[256 \rightarrow 256 \rightarrow 128]$
  - Activation: ReLU
- **Dueling Streams (Split):**
  - **Value** $V(s)$**:** $128 \rightarrow 128 \rightarrow 1$
  - **Advantage** $A(s,a)$**:** $128 \rightarrow 128 \rightarrow |A|$
- **Output:** $|A| = 4$ (Single) or 8 (Multi)

Training Configuration

- **Optimization:**
  - Optimizer: Adam
  - Learning Rate: 1×10^{-3}
  - Gradient Clip: Max norm 10.0
- **Loss Function:**
  - Huber Loss (Smooth L1)
  - Robust to outliers/noise
- **Stability Mechanisms:**
  - **Double DQN:** Decoupled selection/evaluation
  - **Target Update:** Hard copy every 1000 steps

Aggregation Formula: $Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|}\sum_{a'} A(s,a')\right)$

Hyperparameter Configuration

Optimization Dynamics

- **Learning Rate (α):** 1×10^{-3}
  - Optimizer: Adam (Adaptive moments)
  - Rationale: Balance speed vs. stability
- **Discount Factor (γ):** 0.99
  - Prioritizes long-term flow over immediate rewards
- **Experience Replay:**
  - **Batch Size:** 32 (Parallel processing)
  - **Buffer Size:** $10,000$ transitions

Stability & Exploration

- **ϵ-Greedy Schedule:**
  - Range: $1.0 \rightarrow 0.01$
  - Decay: 0.995 per episode
  - Rationale: High initial exploration
- **Network Stability:**
  - **Target Update:** Every 1000 steps
  - **Gradient Clip:** Max norm 10.0
- **Hidden Layers:**
  - Specs: $[256, 256, 128]$
  - Rationale: Capacity for state complexity without overfitting

Why Dueling Double DQN for Traffic Signal Control?

- **High-Dimensional State Spaces:**
  - Single intersection: 41 dimensions
  - Multi-intersection: 117 dimensions
  - Dueling efficiently learns state values
  - Double DQN provides accurate estimates
- **Complex Action Spaces:**
  - Single: 4 discrete actions
  - Multi: 8 joint actions (2^3 combinations)
  - Dueling separates value vs advantage learning
  - Better generalization across action combinations
- **Safety-Critical Application:**
  - Traffic control affects public safety
  - Double DQN reduces overestimation risks
  - Stable learning prevents dangerous policies
  - Reliable performance in real-time settings
- **Sample Efficiency Requirements:**
  - SUMO simulations are computationally expensive
  - Dueling architecture learns more efficiently
  - Experience replay maximizes sample utilization

SUMO Simulation Environment

Simulator Platform

SUMO (Simulation of Urban MObility)

- Open-source microscopic traffic simulator
- Developed by DLR (German Aerospace Center)
- Industry-standard for traffic research

TraCI Interface

- Real-time control via Python API
- Query traffic state every simulation step
- Set traffic light phases dynamically
- Retrieve vehicle positions, speeds, queues

Network Topology & Infrastructure

Road Network Design

- **Multi-Intersection Corridor:**
  - 3 signalized intersections (C1, C2, C3)
  - Total corridor length: 1000 meters
- **Each Intersection:**
  - 4 approaches (N, S, E, W)
  - 3 lanes per approach (12 lanes total)
- **Road Specifications:**
  - Speed limit: 50 km/h (13.89 m/s)
  - N/S exit distances: 200 meters

Traffic Light Configuration

- **Phase Structure:**
  - Phase 0: North-South green
  - Phase 1: North-South yellow(3s)
  - Phase 2: East-West green
  - Phase 3: East-West yellow(3s)
- **Timing Constraints:**
  - Yellow duration: 3s (fixed)
  - Red clearance: 2s
  - Minimum green: 5s
  - Maximum green: 60s
- **Agent Control:**
  - Selects green phases only
  - Decision frequency: Every 5 seconds

Traffic Generation & Demand Patterns

Vehicle Characteristics

- **Vehicle Types:**
  - Standard cars: 90% of traffic
  - Buses: 10% (longer, slower)
- **Car Parameters:**
  - Length: 5.0 meters
  - Max speed: 60 km/h (16.67 m/s)
  - Acceleration: 2.6 m/s^2
  - Deceleration: 4.5 m/s^2
- **Bus Parameters:**
  - Length: 12.0 meters
  - Max speed: 50 km/h (13.89 m/s)
  - Acceleration: 1.2 m/s^2
  - Deceleration: 4.0 m/s^2

Demand Profiles

- **Time-Varying Traffic:**
  - Period 1 (0-900s): Low traffic
  - Period 2 (900-1800s): Medium traffic
  - Period 3 (1800-2700s): Peak traffic
  - Period 4 (2700-3600s): Declining
- **Flow Probabilities:**
  - Low: 0.02 vehicles/second
  - Medium: 0.03 vehicles/second
  - Peak: 0.04 vehicles/second

Simulation & Episode Configuration

Episode Structure

- **Duration:** 3600 sec (1 h)
  - Captures full demand cycle
  - Includes congestion buildup/dissipation
- **Decision Frequency:** Every 5 sec
  - 720 decision steps per episode
  - Balance responsiveness vs stability
  - Matches practical deployment constraints
- **Training Episodes:**
  - Range: 200-500 episodes
  - Varies by hyperparameter configuration

Metrics & Evaluation

- **Primary Metrics:**
  - **Waiting time:** Total time vehicles spend halted (seconds)
  - **Queue length:** Number of stopped vehicles
  - **Throughput:** Vehicles completing journey

Systematic Hyperparameter Ablation Study

Fixed Parameters

- Architecture: [256, 256, 128]
- Target network update: every 1000 steps
- Optimizer: Adam + gradient clipping (10.0)
- Loss: Smooth L1 (Huber)
- Episode length: 3600s (720 decisions)
- ϵ-greedy: $1.0 \rightarrow 0.01$

Parameters Varied

- **Learning rate (α):** 0.0001 – 0.002 (stability vs convergence speed)
- **Discount factor (γ):** 0.95 – 0.995 (short-term vs long-term planning)
- **Epsilon decay:** 0.98 – 0.999 (exploration–exploitation balance)
- **Batch size:** 64, 96, 128 (gradient estimate stability)
- **Replay buffer size:** 50k, 75k, 100k (experience diversity vs staleness)

12 Experimental Configurations

Run	LR	γ	Batch	ϵ Decay	Buf.	Eps
01	0.0005	0.99	64	0.995	50k	500
02	0.0001	0.99	64	0.995	50k	500
03	0.001	0.99	128	0.995	50k	500
04	0.002	0.99	128	0.995	50k	500
05	0.0005	0.99	64	**0.999**	50k	500
06	0.0005	0.99	64	**0.99**	50k	500
07	0.0005	0.99	64	**0.98**	50k	200
08	0.0005	**0.95**	64	0.995	50k	200
09	0.0005	**0.995**	64	0.995	50k	200
10	0.0001	0.99	128	0.999	**100k**	500
11	0.001	0.95	64	0.99	50k	500
12	0.0003	0.99	**96**	0.997	**75k**	500

- **Runs 01-04:** LR sensitivity,
- **05-07:** ϵ decay ,
- **08-09:** γ effects

Multi-Intersection Performance Results (Part 1)

Configuration	Wait (s)	Queue	Thru.
Baselines			
Fixed-Time	660,034	589	429
Actuated	618,820	750	544
Top RL Configs			
Run 01	**1,283**	**83**	**565**
Run 05	11,069	203	659
Run 10	6,430	133	682
Run 02	18,940	254	786
Run 04	15,221	227	514

Best: Run 01

- **99.8%** wait reduction
- **85.9%** queue reduction
- LR=0.0005, γ=0.99, ϵ_{decay}=0.995

Key Findings

- Slow ϵ decay critical
- Moderate LR (0.0001-0.0005)
- 100$\times$ better than baseline

Failed Configurations & Lessons

Config	Wait (s)	Improvement
Fixed-Time	660,034	–
Run 06 (ϵ=0.99)	443,071	32.9%
Run 09 (γ=0.995)	357,916	45.8%
Run 08 (γ=0.95)	161,072	75.6%
Run 03 (LR=0.001)	62,146	90.6%
Run 07 (ϵ=0.98)	25,707	96.1%

Critical Failures

- **Run 06:** Fast ϵ (0.99)
  - Premature exploitation
  - Only 33% improvement
- **Run 09:** High γ (0.995)
  - Unstable value estimates
  - Only 46% improvement

Key Lessons

- Fast ϵ decay catastrophic
- Extreme γ hurts
- Proper tuning essential

Critical: Epsilon Decay Dominates Performance

Impact

Decay	Run	Reduction
0.999	05, 10	**98-99%**
0.995	01-04	90-99.8%
0.99	06	32.9%

Why Critical?

- Delayed feedback through network
- Multi-modal policies
- $8^3 = 512$ joint actions to explore

Exploration Timeline

Slow (0.999) ✓

Ep 100: ϵ=0.90, Ep 500: ϵ=0.61
Result: 98-99% improvement

Fast (0.99) ×

Ep 100: ϵ=0.37, Ep 500: ϵ=0.01
Result: Only 33% improvement

Use ϵ decay $\geq$ 0.995

Discount Factor: Non-Monotonic Behavior

γ	**Horizon**	**Result**
0.95	100s	75.6%
0.99	**500s**	**95-99.8%**
0.995	1000s	45.8%

Too Low (0.95)

- Myopic, reactive only
- Misses future congestion
- 75.6% (mediocre)

Optimal (0.99)

- Balances near/far future
- Captures traffic waves
- 95-99.8% (excellent)

Too High (0.995)

- Excessive long-term focus
- Bootstrapping errors accumulate
- Credit assignment fails
- Training unstable
- Only 45.8% (failure!)

Use $\gamma = 0.99$

Learning Rate & Batch Size Interactions

LR	Run	Result
0.0001	02, 10	97-99%
0.0005	**01, 05**	**98-99.8%**
0.001	03	90%
0.002	04	97.7%

Insights

- Conservative (0.0001): Stable, slower
- Moderate (0.0005): Best balance
- Aggressive: Needs large batch

High LR + Large Batch

Run 04: LR=0.002, Batch=128
97.7% - *batch stabilizes*

High LR + Wrong γ

Run 03: LR=0.001, Batch=128
90.6% - *interactions matter*

Batch Findings

- 64: Best for moderate LR
- 128: Needed for high LR

Use LR=0.0005, Batch=64

Critical: Early Stopping Essential

Best vs Final (Run 01)

Model	Ep	Wait (s)
Best	97	**1,283**
Final	500	214,763
Degrade		$\times$167!

Run	Best Ep
01	97
02	186
05	167
10	27

What Failed?

- Training past optimal point
- Overfits to training patterns
- Best at episodes 27-186
- Never at final episode!

Why Needed?

- Memorizes training seed
- Poor generalization
- Validation plateaus early

Performance Comparison with Baselines

Baseline Controllers

- **Fixed-Time Control:**
  - Pre-programmed phase durations
  - Through/right: 30s green
  - Left turn: 15s green
  - Total cycle: 110 seconds
- **Actuated Control:**
  - Vehicle-actuated phases
  - Min green: 5s, Max: 60s
  - Extension: 3s per detection
  - Gap threshold: 3.0s
- **Dueling Double DQN (Ours):**
  - Learned adaptive policy
  - Network-wide coordination
  - Optimizes cumulative reward

Multi-Intersection Results

Method	Wait (s)	Throughput
Fixed-Time	660,034	429
Actuated	618,820	544
DQN Best	**1,283**	**565**
Improvement	**99.8%**	**+31.7%**

Single Intersection:

Method	Wait (s)	Queue
RL-DQN	488	75
Fixed-Time	3,523	101
Actuated	5,656	143
Improvement	**86-91%**	**26-48%**

Key Experimental Findings

- **Exploration is Critical:**
  - Slow ϵ decay ($\geq$0.995): 95-99.8% wait reduction
  - Fast decay (0.98-0.99): Only 18-33% reduction
  - Extended exploration essential for multi-intersection coordination
- **Discount Factor Sensitivity:**
  - $\gamma = 0.95$ (myopic): 75.6% reduction
  - $\gamma = 0.99$ (optimal): 95-99.8% reduction
  - $\gamma = 0.995$ (unstable): Only 45.8% reduction
- **Early Stopping Matters:**
  - Best checkpoints: Episodes 27-186
  - Final models (Ep 500): Up to 167$\times$ worse!
  - Overfitting to training distribution
- **Multi-Intersection Coordination:**
  - Centralized agent learns implicit coordination
  - Best config: 65.5% queue reduction, 53.6% throughput gain
  - No explicit communication needed

Policy Demonstration Video

Click here to watch the video

Trained DQN agent controlling multi-intersection traffic in SUMO

Summary & Key Findings

- **Problem:** Optimized multi-intersection traffic signal control using Deep RL
- **Approach:**
  - Formulated as MDP with high-dimensional state space (117D)
  - Implemented Dueling Double DQN for stable learning
  - Single agent controls network-wide coordination
- **Results:**
  - **98.3%** reduction in waiting time vs fixed-time (11,069s vs 660,034s)
  - **65.5%** reduction in queue length (203 vs 589 vehicles)
  - **53.6%** improvement in throughput (659 vs 429 vehicles)
  - Stable convergence with proper hyperparameters
- **Key Insights:**
  - Slow ϵ decay ($\geq$0.995) is critical for success
  - Early stopping prevents overfitting (167$\times$ degradation observed)
  - Optimal γ=0.99 balances myopic vs unstable learning
  - 12-config ablation revealed hyperparameter interactions

Questions?

Thank you for your attention!

Abdul Basit & M. Abdullah
LUMS