

---

# **Smart Emergency Traffic Control**

---

**Zakarya Boudraf** (Matricola: 0522501649)

Information Visualization - Master's in Internet of Things - University of Salerno

**Instructors:** Prof. Lucia Cimmino, Prof. Andrea Francesco ABATE

# Context & Motivation (The "Why")

- 🏢 **The Problem:** Traditional "dumb" fixed-timer lights fail to adapt to real-time traffic chaos.
- 之心 **The Risk:** In medical emergencies, every second counts, gridlock equals mortality.
- 💡 **The Solution:** Moving to an AI agent that "sees" ambulances and clears paths intelligently.



# Project Evolution & Objectives

## Original Scope

Initially designed to manage a single isolated intersection using basic heuristics.

**Challenge:** Isolated nodes cannot simulate sophisticated network behaviors like coordinated flow.

## Academic Guidance

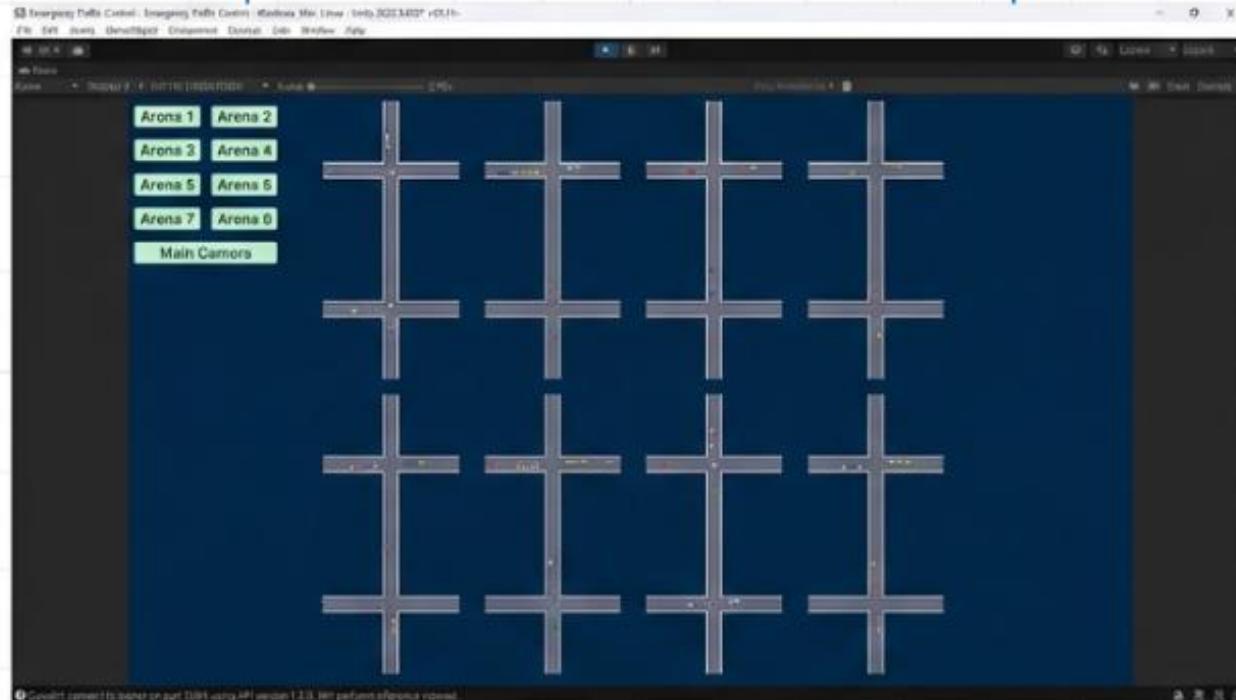
Prof. Cimmino recommended expanding to a **dual consecutive intersection topology**.

**Goal:** Force the agent to learn spatial-temporal coordination and predictive arrivals.

# System Architecture

Engine: Unity 2022.3.62f2  
+ ML-Agents 2.0.2

Algorithm: Proximal Policy Optimization (PPO)



Parallel Training Environment (8x Speed)

Architecture: 8 Simultaneous Arenas. This parallelization allows the agent to collect 8x the experience per second, stabilizing learning against stochastic traffic generation.

# Implementation Challenges

- ⚠ **Dependency Conflicts:** High-friction versioning between protobuf, numpy, and tensorflow-intel.
- 💡 **Isolated Environments:** Required building custom Python virtual environments to bridge Unity C# and Python backends.
- 💡 **The Solution:** Strict version-alignment to stabilize the bridge between simulation and training.



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window displays several error messages related to dependency conflicts between tensorflow-intel, protobuf, typing-extensions, and numpy. The errors indicate incompatible versions for tensorflow-intel (~2.13.0) and numpy (~1.21.2). It also shows a warning about pip version (~21.1.1) and a stack trace for an ImportError involving tensorflow.compat and tensorflow.python.libcompat. At the bottom, it shows version information for ml-agents (~0.20.0), ml-agents-envs (~0.20.0), Communicator API (~1.5.0), and PyTorch (~2.0.1+cpu).

```
tensorflow-intel 2.13.0 requires protobuf<=8.21.0,!=8.21.1,!=8.21.2,!=8.21.3,!=8.23.0,!=8.23.5,<8.0.0dev,==2.20.3, but you have protobuf 3.20.0 which is incompatible.
tensorflow-intel 2.13.0 requires typing-extensions<4.0.0,>=3.6.6, but you have typing-extensions 4.12.2 which is incompatible.
numpy 0.58.1 requires numpy<1.20,>=1.22, but you have numpy 1.21.2 which is incompatible.
Successfully installed numpy-1.21.2
WARNING: You are using pip version 21.1.1; however, version 20.0.1 is available.
You should consider upgrading via the 'c:\users\zakarya\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
PS C:\Study\Information Visualization\Wadty Projects\Emergency Traffic Control> cd "C:\Study\Information Visualization\Unity Projects\Emergency Traffic Control"
PS C:\Study\Information Visualization\Wadty Projects\Emergency Traffic Control> ml-agents-learn Assets/traffic.yaml --run-id=EmergencyRun1 --force
Traceback (most recent call last):
  File "c:\users\zakarya\appdata\local\programs\python\python38\lib\site-packages\tensorboard\compat\_init_.py", line 42, in tf
    from tensorboard.compat import notf # noqa: F401
ImportError: cannot import name 'notf' from 'tensorboard.compat' (c:\users\zakarya\appdata\local\programs\python\python38\lib\site-packages\tensorboard\compat\_init_.py)

During handling of the above exception, another exception occurred:

RuntimeError: module compiled against API version 0xf but this version of numpy is 0xe

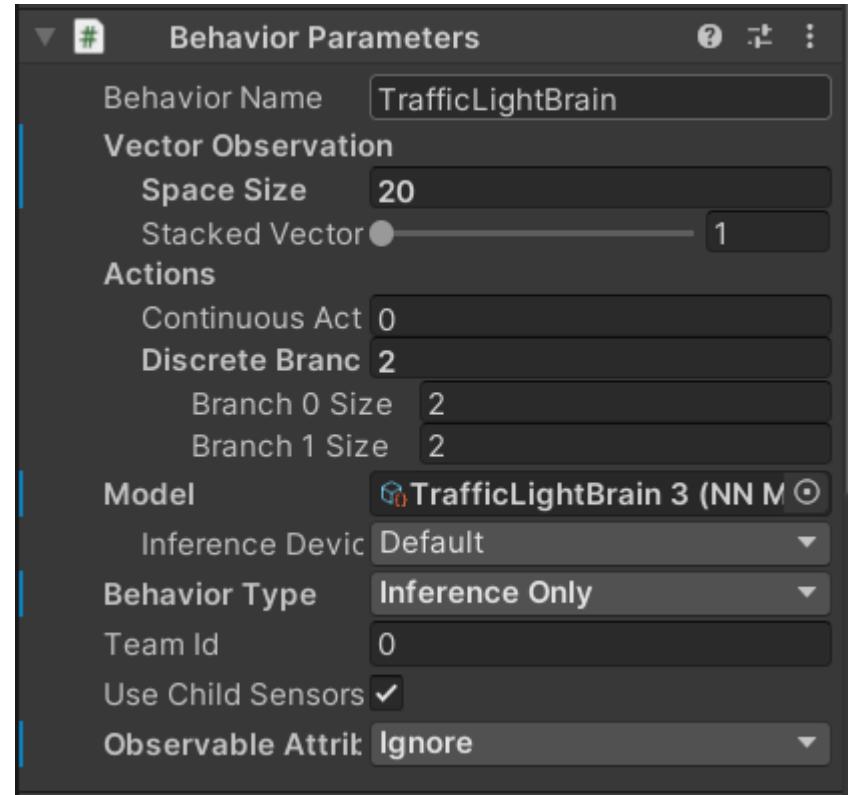
Version Information:
ml-agents: 0.20.0
ml-agents-envs: 0.20.0
Communicator API: 1.5.0
PyTorch: 2.0.1+cpu
```

# Agent Perception (The "Eyes")

The agent "sees" through a **Vector of 20 observations:**

**1. Intersection Data (16 obs):** This is the raw sensor data, consisting of discrete counts of cars and ambulances waiting at each of the eight approaches across the two intersections

**2. Internal State (4 obs):** The agent possesses self-awareness of its own state, observing the current light phase



# Safety Constraints

## Multi-Discrete Control

The agent outputs simultaneous switch/hold actions for both intersections. However, neural networks can be erratic.

**The Safety Layer:** I implemented a hard-constraint in C# to enforce a minimumGreenTime. This prevents dangerous "flickering" and ensures realism.

After some experimentation the timer was set to **2 seconds** as the sweet spot to prevent "flickering" while maintaining freedom to switch when necessary.

```
if (action_I1 != i1_LightState) {  
    if (timeSinceLastSwitchI1 >= minimumGreenTime) {  
        i1_LightState = action_I1; // Allowed  
    } else {  
        // ACTION BLOCKED  
    }  
}
```

# Reward (or Penalty) Engineering

$$R_{\text{total}} = R_{\text{flow}} + R_{\text{emergency}} + R_{\text{coordination}}$$

## Efficiency

-0.001 Penalty per step for each waiting car.

Reason: To constantly flush queues.

## Priority

+2.0 Reward for clearing an Ambulance.

-2.0 Penalty if an Ambulance stops.

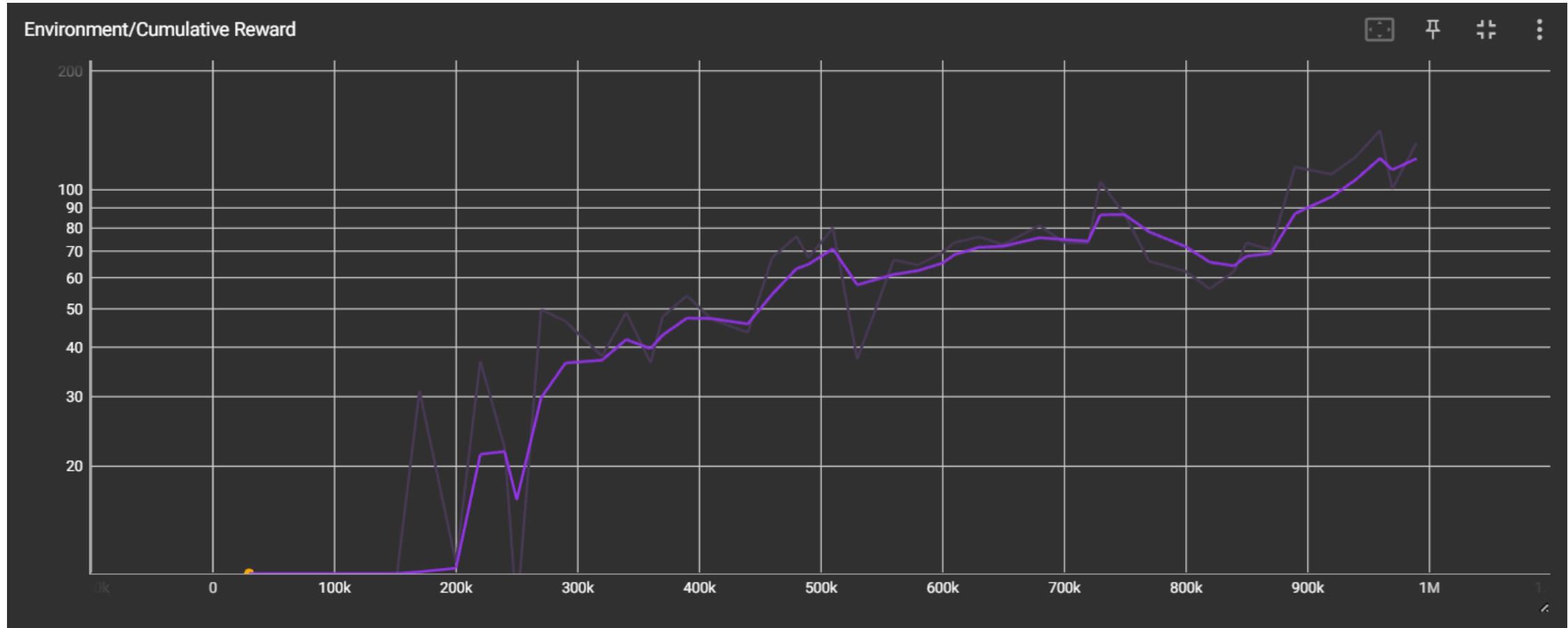
Result: The 'Green Wave' is learned.

## Emergent Behavior

The agent learned that to secure the +2.0 at Intersection 2, it must open Intersection 1 early.

This is learned temporal planning.

# Quantitative Results (Training)

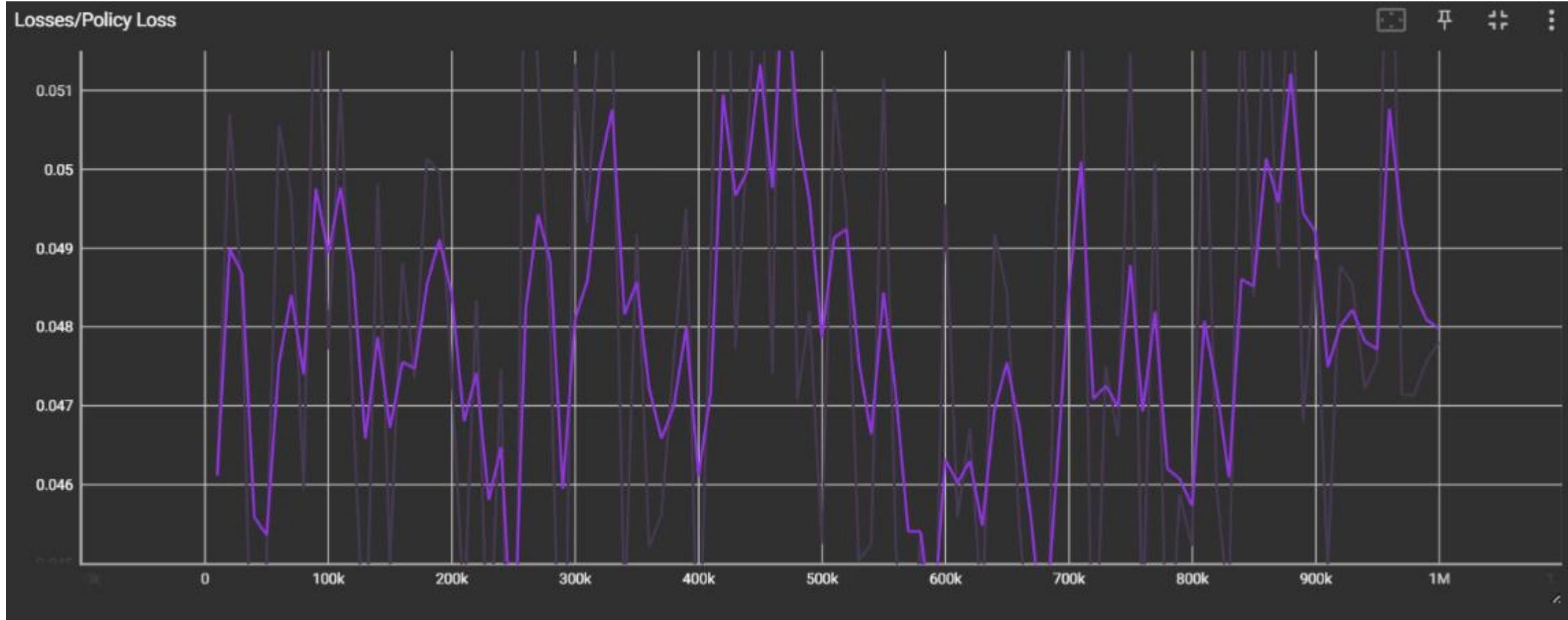


Mean Reward: ~140

Convergence: 700k

Policy Loss: ~0.005

# Quantitative Results (Training)



Mean Reward: ~140

Convergence: 700k

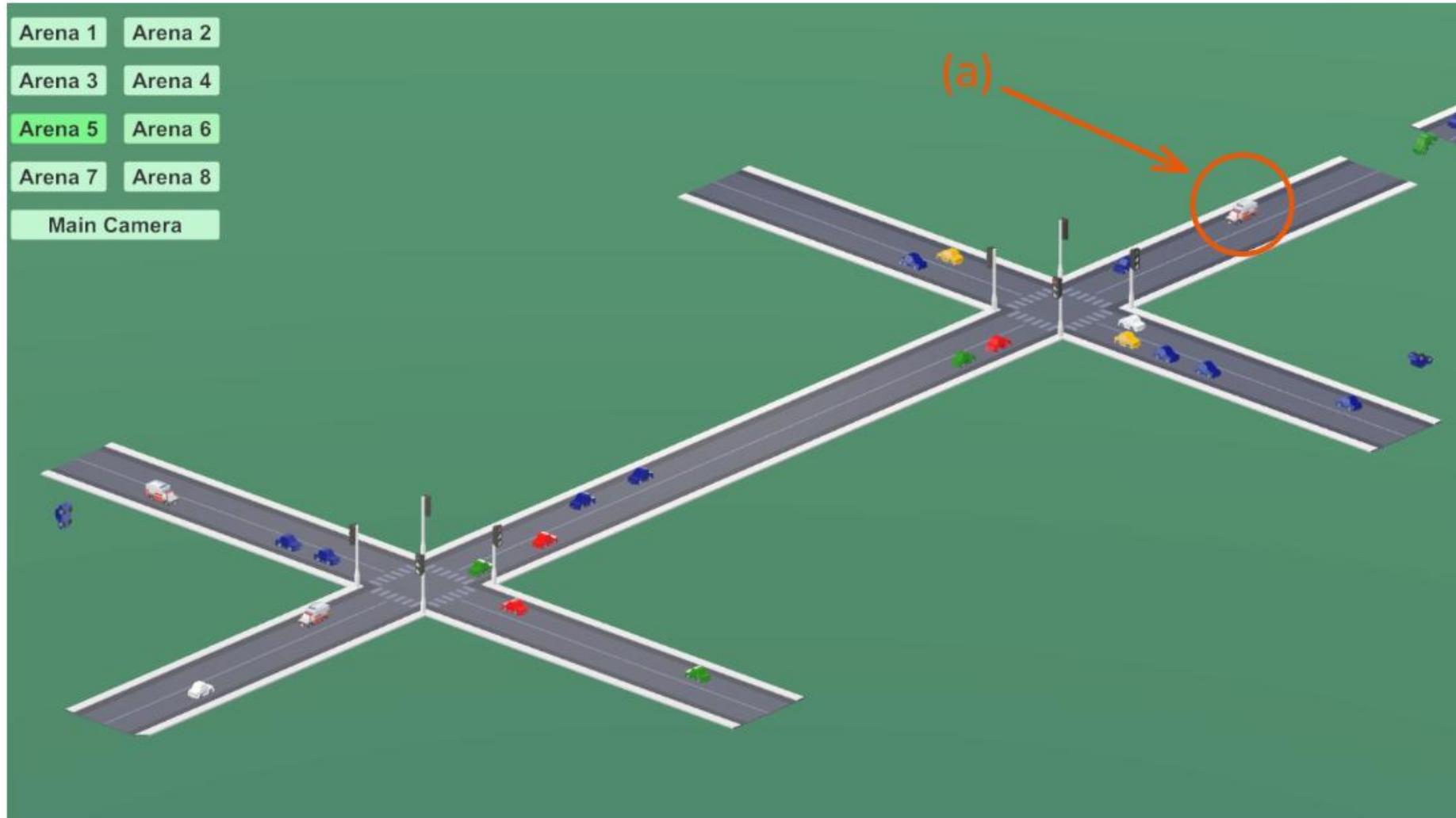
Policy Loss: ~0.005

# The "Green Wave" Result

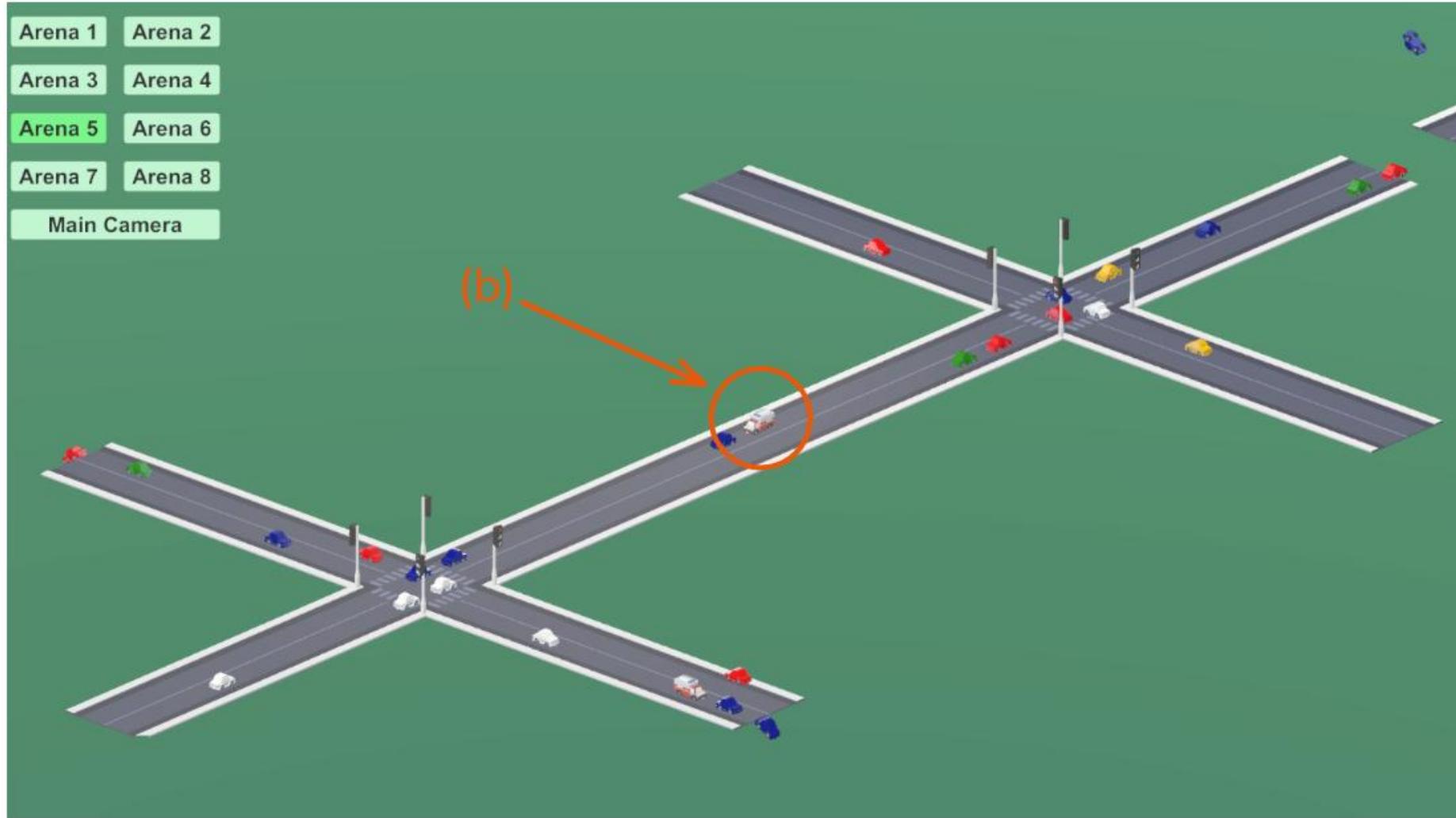
Ambulance stops were reduced from 85% (Fixed Timer) to <5%  
(AI-Powered).

The agent anticipates arrival across the bridge, turning Node 2 green before arrival.

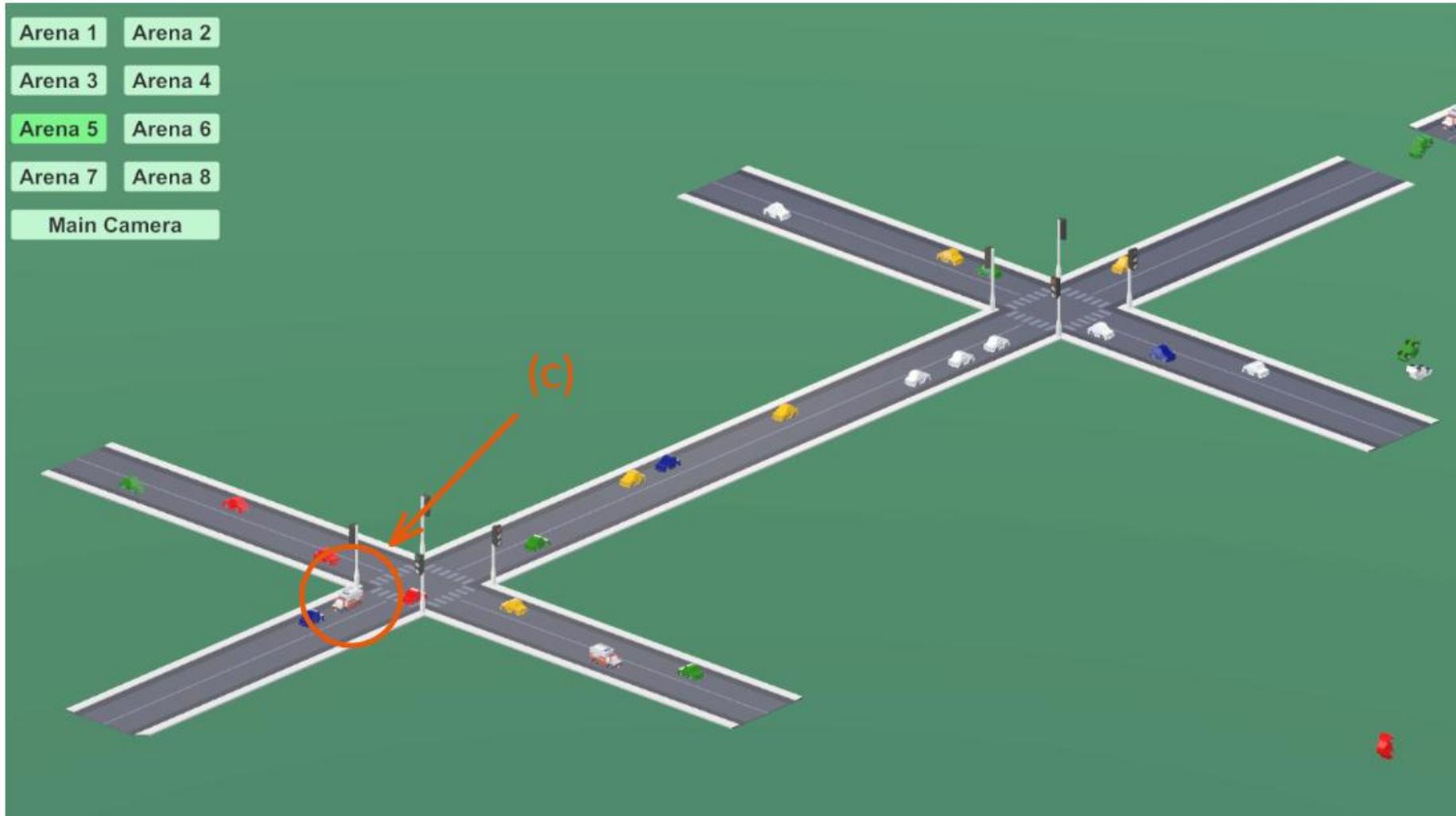
# The Green Wave



# The Green Wave



# The Green Wave



# Interactive Visualization Features



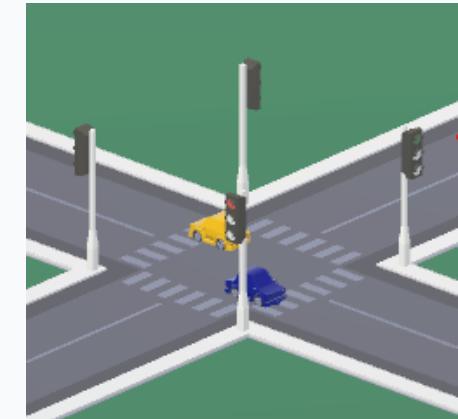
## Camera Switcher

Toggle between **Strategic**  
**Top-down View** and  
**Close-up Intersection**  
View for detailed analysis.



## Traffic Light Visuals

In the close-up intersection  
view the traffic lights  
switch in real time which  
allows inspection of model  
behavior



## Conclusion & Future Work

This project serves as proof that **reinforcement learning** allows for transforming a static infrastructure into a dynamic, city-scale cognitive network capable of real-time, multi-objective optimization.

-  **Complexity:** Adding yellow lights as well as car turns will

-  be imperative for implementation in real life settings.

-  **Scalability:** Ready for deployment across large arterial

-  corridors.

**Thank You! | Questions?**