Machine Learning Nanodegree

Capstone Proposal v2

Traffic Simulation / Optimization

Joseph Kessler

joe@kessler.net

Contents

Domain Background

Impact of Inefficient Traffic Management

An area of interest to me has always been traffic intersections. All of us have been stopped at intersections for no apparent reason – often when there nobody else is even nearby. This causes frustration, increases commute times, and contributes to fuel costs. The main reason is the relatively simple sensor/timing systems employed by current intersection controllers.

ML to the Rescue!

Fortunately, new technologies have arrived that could make the transit process more efficient, cost effective, and less frustrating if they are applied in an effective manner.

Other Similar Efforts

Traffic simulation and optimization is a problem area where there has been considerable prior work, using different approaches. In most of those cases the intention, like mine, was to improve throughput and thereby attain the side benefits of doing so. My simulation will be relatively simple compared to some of these.

University of Hartford

University of Hartford studied a series of intersections on how alignment of signal timing could result in reduced vehicle delays and stops. This resulted in some concrete recommendations made to the city.

http://www.hartford.edu/ceta/about-us/facultystaff/ceb/fang/student-projects/traffic-simulation.aspx

University of Hartford

ResearchGate

There are papers available regarding studies of traffic flow through intersections.

https://www.researchgate.net/publication/260197818_Optimization_of_a_complex_urban_intersection_using_discrete_event_simulation_and_evolutionary_algorithms

https://www.researchgate.net/publication/261165366_Agent-based_traffic_simulation_and_traffic_signal_timing_optimization_with_GPU

The IEEE

IEEE also offers many papers on this same topic.

Commercial Traffic Simulation Software

Commercial software also exists, such as from AnyLogic:

https://www.anylogic.com/road-traffic/

Problem Statement

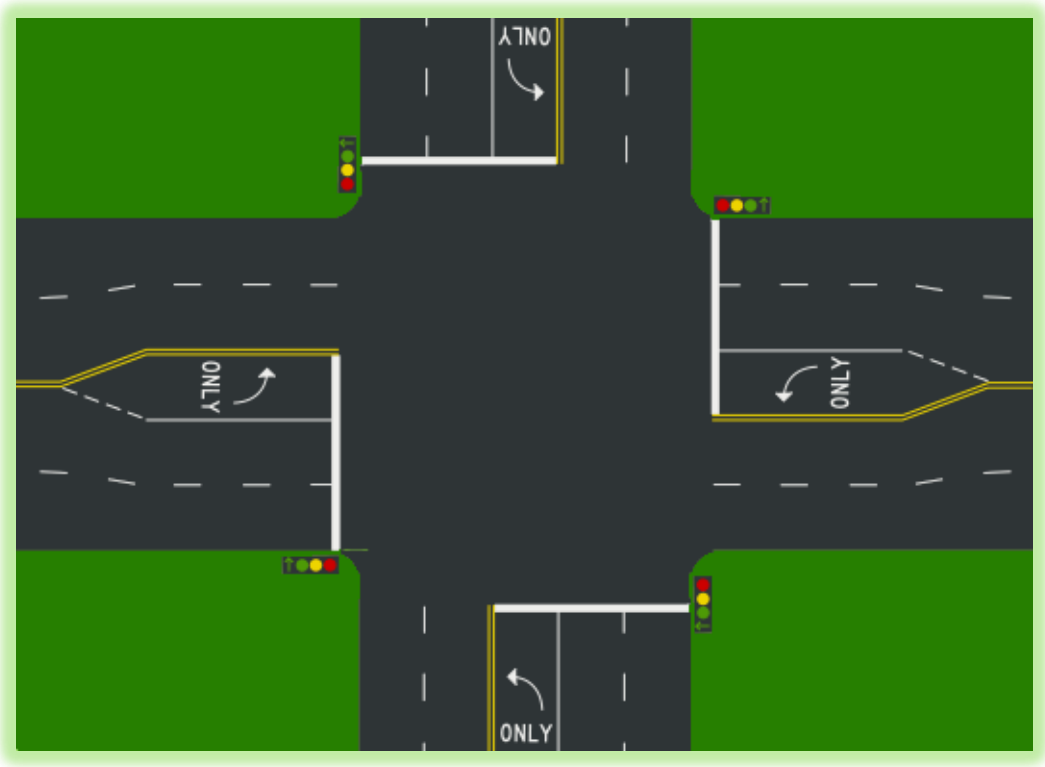
What We are Trying to Achieve

The problem presented is: how can we optimize the behavior of traffic lights to maximize several key measures related to the vehicles traveling through those intersections? At a slightly higher scale, can we apply those behaviors to several intersections in a small fictional neighborhood to observe the effects at that level as well?

The Model

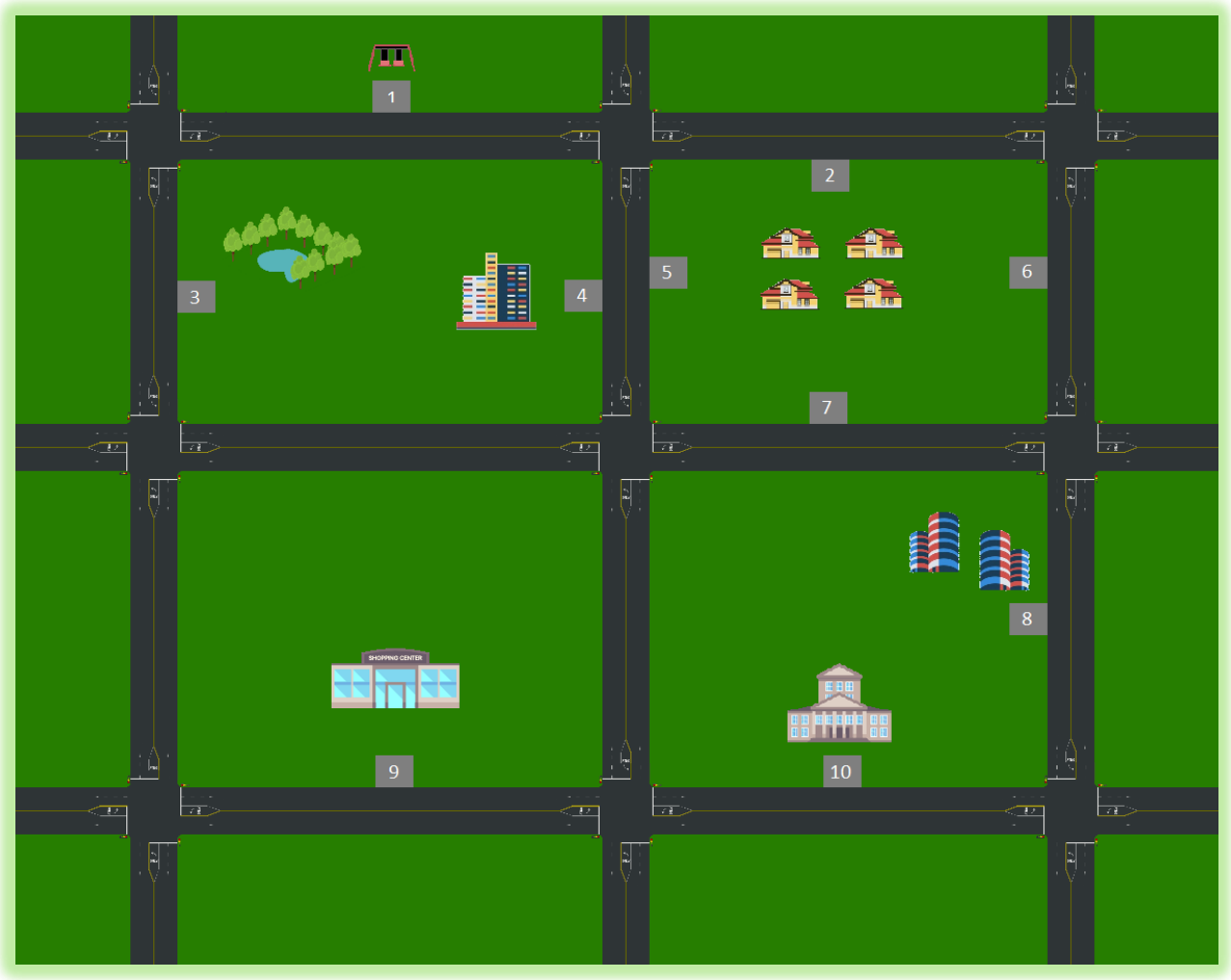
Intersection Layout

We will model three things: simple traffic flow, a traffic intersection, and a small fictional neighborhood containing several of those items working in parallel to assess a somewhat broader impact. All the individual intersections will have the same layout:



The Neighborhood

At a higher level, a fictional neighborhood consisting of several intersections and destinations will be modeled, looking like this:



Each car will travel into the neighborhood, or start from somewhere in the neighborhood, to a specific target and may leave the neighborhood (after which it is no longer tracked). Probability of certain destinations will vary based upon the time of day.

Traffic Volume

Some road will have heavier traffic in general, and the overall traffic will reflect patterns like what we typically experience for of time-of-day, day-of-week, etc.

Constraints

For each inbound direction, we will model these possible states:

- Red, green,
- Red-left-arrow or green-left-arrow.

To keep the simulation simple, we will not include yellow. Other constraints include:

- At any given time, a green light can only appear for one direction and the rest will be red.
- It cannot “starve” any oncoming traffic for more than 2 minutes.

The problem domain does not include consideration for inclement weather, road construction, or accidents.

Datasets and Inputs

Data for the Agent

The agent (intersection) will receive the following information from sensors and the environment in general:

- Current date/time
- Sensor states at the stop lines
- Visual images coming from each oncoming direction

Data for the Environment

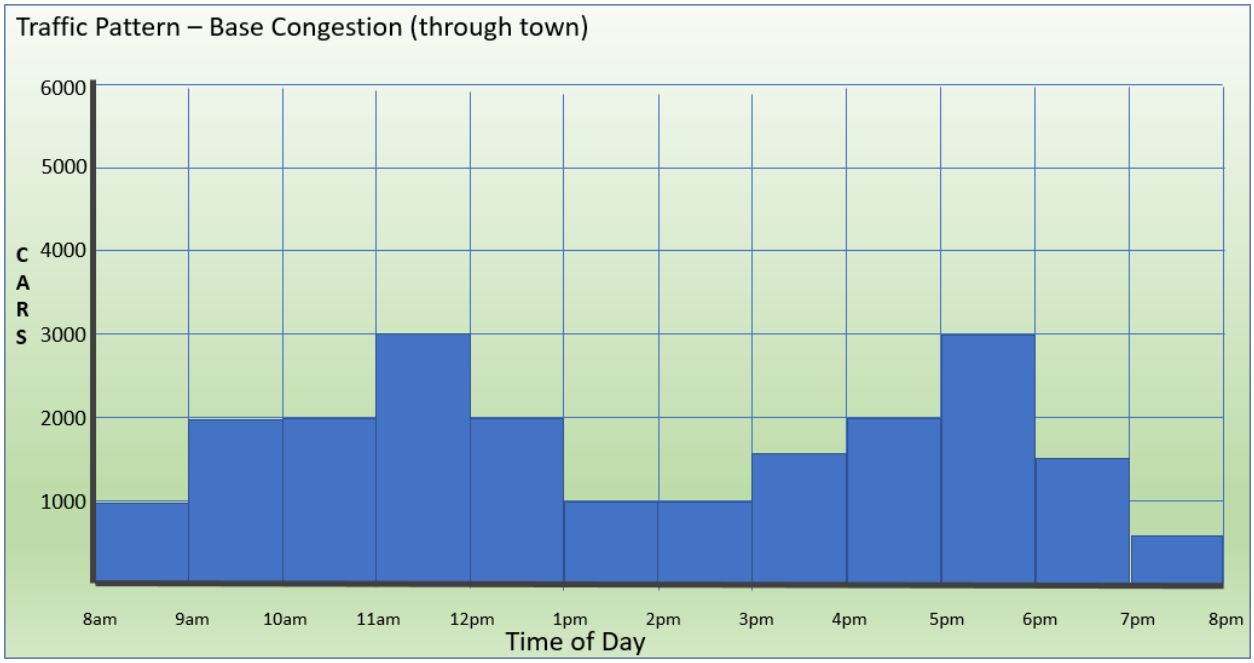
This problem is being defined as a reinforcement learning problem combined using discrete event simulation. There are two models that must be maintained for this complex simulation – one for the agent, and one for the environment.

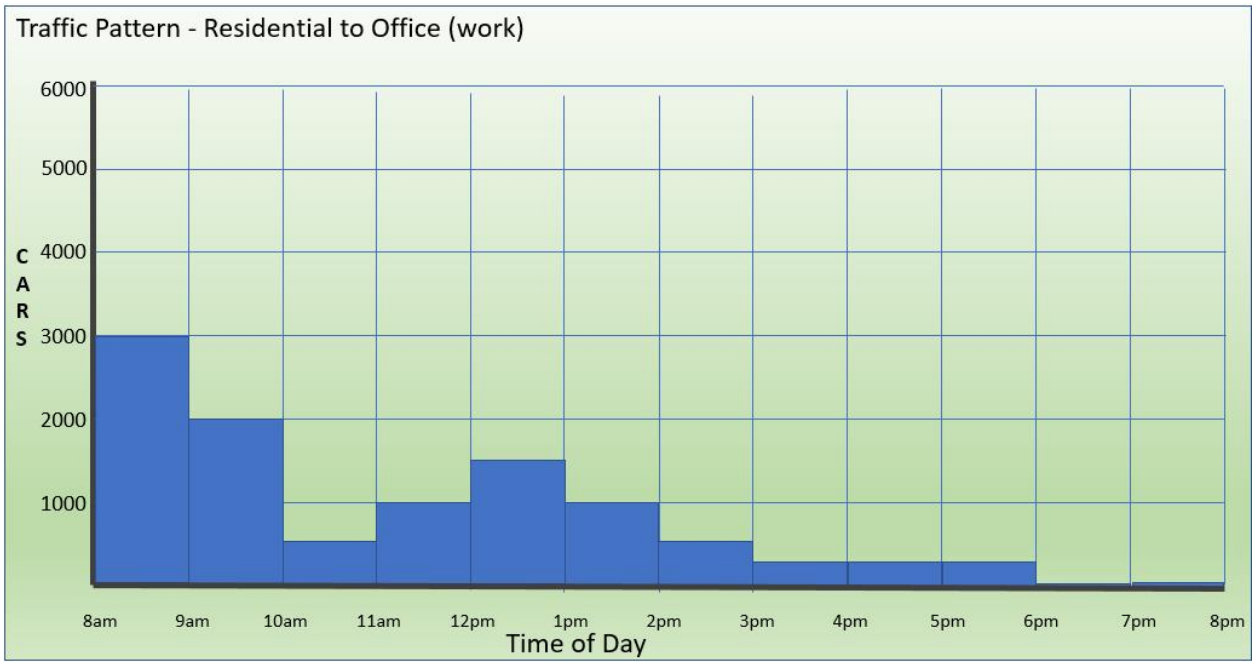
The environment must model:

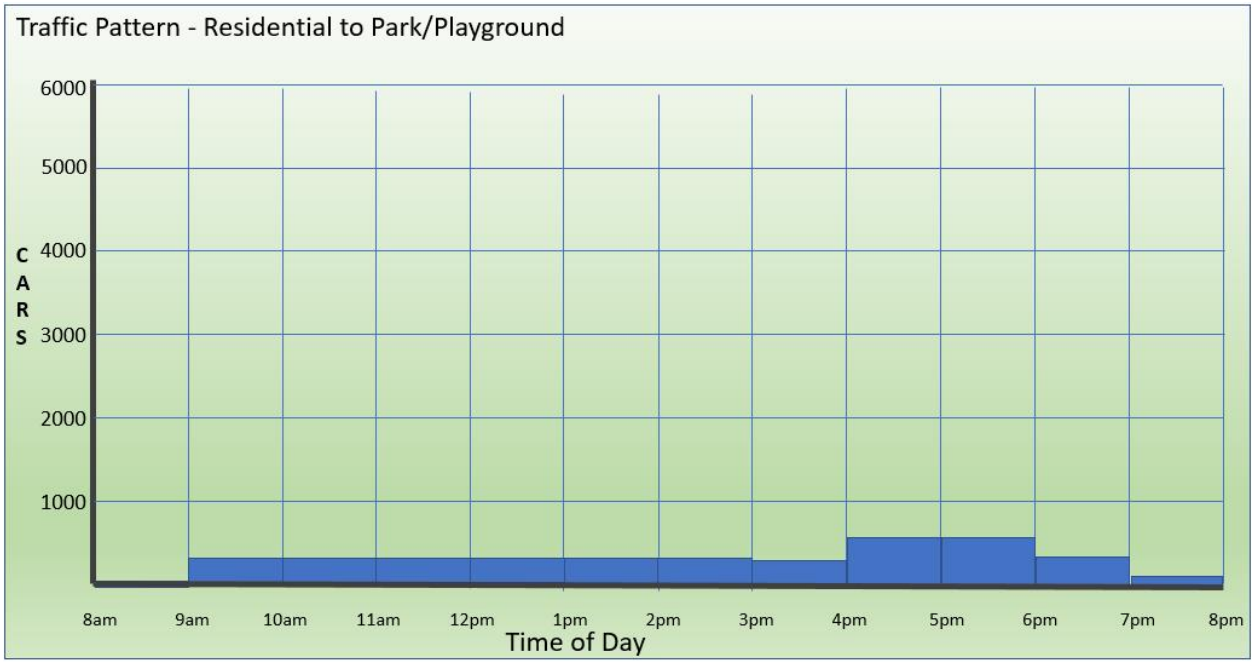
- Traffic volume and likely destinations at a variety of times,
- Sensor information for sensors at each stop line, and
- Visual information that will model the appearance of oncoming cars (camera sensor)

Fictional training data will be generated for the environment.

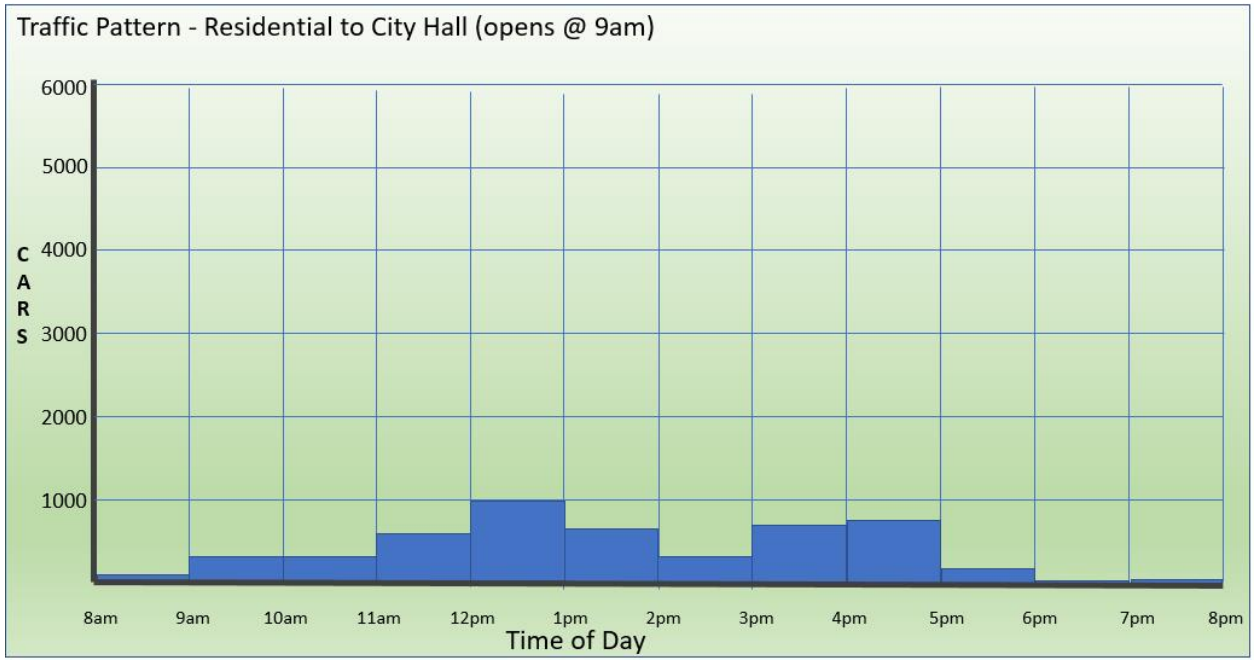
Traffic Patterns Between Destinations

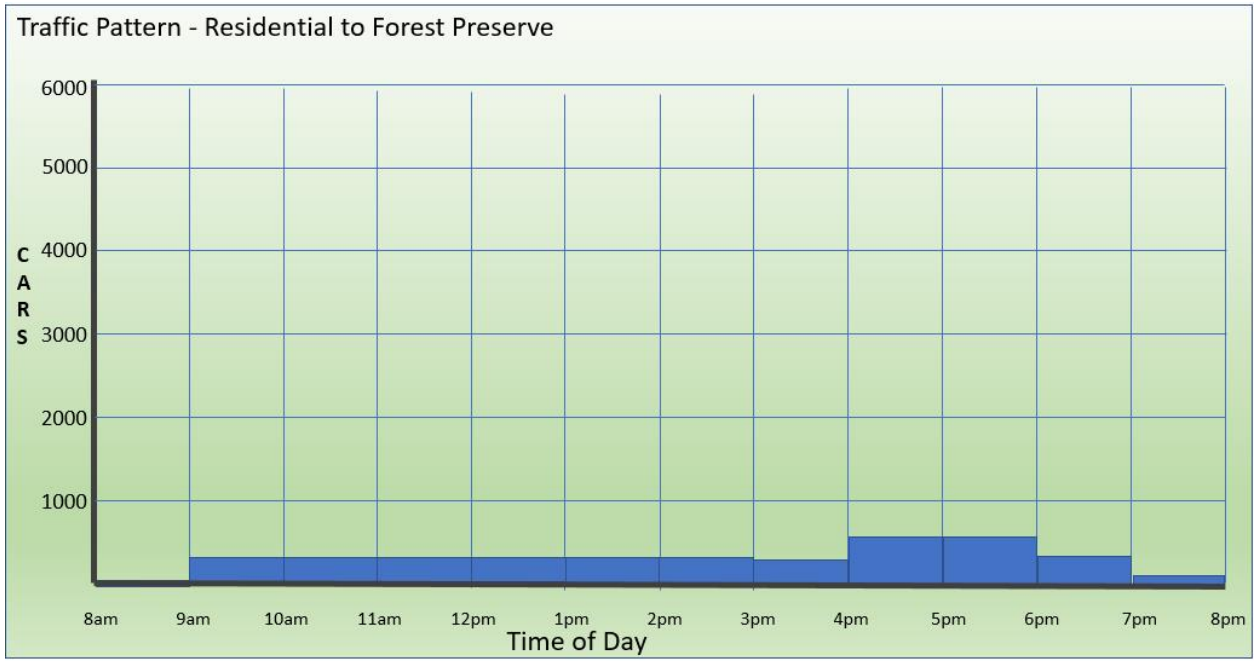



Traffic Pattern - Residential to Office (work)
6000
5000
4000
3000
2000
1000
CARS
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day


Traffic Pattern - Residential to Park/Playground
6000
5000
4000
3000
2000
1000
CARS
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day


Traffic Pattern - Residential to Shopping Mall (opens @ 10am)
6000
5000
4000
3000
2000
1000
CARS
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day


Traffic Pattern - Residential to City Hall (opens @ 9am)
6000
5000
4000
3000
2000
1000
CARS
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day



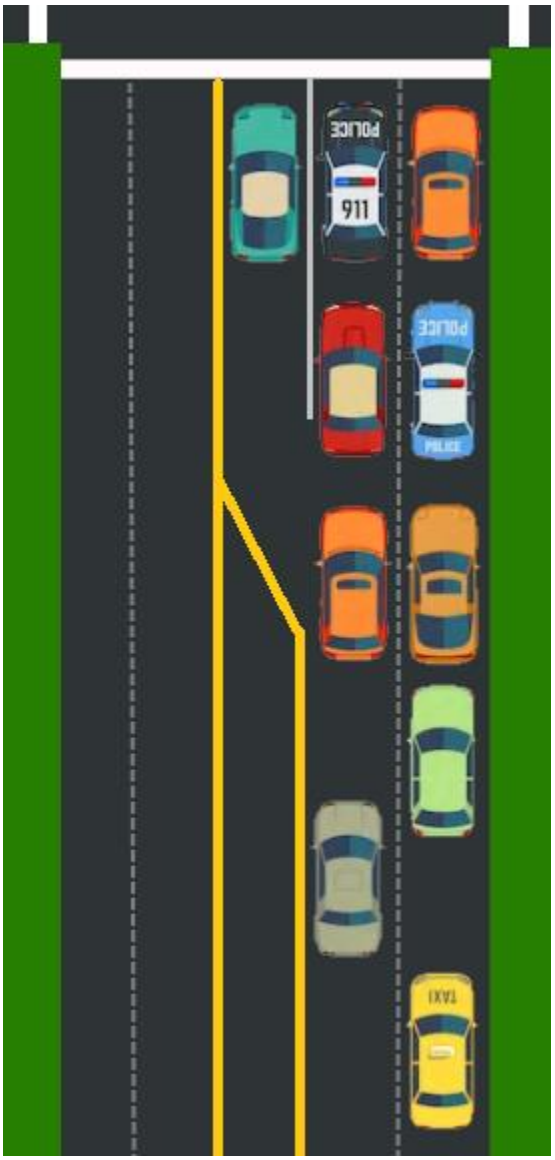
Traffic Bursts

There will be periodic “bursts” of traffic to test the agents’ ability to handle unforeseen changes to the general pattern.

Format of the Data

The environment will stochastically generate simulated “cars” based upon the traffic patterns described. Each car will be tracked from source to destination.

In addition to simulating cars, the environment will generate “video” snapshots that simulate camera hardware at the intersection pointing at each approaching direction. The images will be generated based upon the location of cars relative to the intersection. To keep things simple, we will replace a more realistic perspective view with an overhead view of approaching traffic:



This is an example but, in the simulation, the agent will be able to see roughly ¼ mile of oncoming traffic – giving it time to incorporate that information into its decision making.

Solution Statement

The solution will involve discrete event simulation of both:

- Simple traffic flow and rules (behavior cars in the environment)
- Traffic lights (agents) being trained for efficiency.

Traffic flow simulation will be implemented in the environment including observing the rules of traffic.

Visual information from cameras will be processed using a CNN. It will be able to determine the presence of oncoming cars that are not yet at the stop line sensors, allowing it to potentially change states in anticipation – or not, if appropriate.

The solution will not assume that additional sensors are place under the road but will instead use (virtual) cameras mounted and facing the four oncoming directions.

Benchmark Model

A benchmark will be developed by implementing a "stock" behavior that all the intersections (agents) will employee. This default behavior of a light is described here. Assuming that an intersection is at an all-stop condition, and the driver is going North..

```
Do indefinitely
    Case(turn lane sensors)
        Northbound turn lane occupied and southbound empty:
            Enable northbound's left green arrow (allowing them to turn west)
            Enable northbound's main green light (allowing movement north)

        Northbound empty and southbound turn lane occupied:
            Enable southbound's left green arrow (allowing them to turn east)
            Enable southbound's main green light (allowing movement south)

        Turn lanes empty in both directions:
            Keep lane sensors red in both directions
            Enable both sets of main green lights (allow north/south traffic)

        Turn lanes occupied in both directions:
            Enable green arrows on both sides (but main lights are red)

    Maintain current state 10 seconds (max left-turn time or base for main green)
    Turn off any left arrows (set to red)
    Enable main green lights and allow northbound and southbound travel

    Maintain state for 15 seconds (minimum for open north/south traffic)

    Trigger all stop (all lights red in all directions)
    Continue same logic, this time switching to east-west
```

The elements highlighted in **green** are the actual areas that we are trying to optimize!!!

One simple thing to note is that, based upon this logic, cars sitting at an intersection (and not in a turn lane) will get at least 15 seconds of main green arrow. They will also get 10 more seconds unless the opposite side has cars waiting to turn left.

Metrics from this "standard" logic will be used as the baseline to measure performance of the ML-based enhancements. Note that the agent here does not use a video feed, much like most real intersections today. (Red-light cameras exist but those serve a different purpose)

Evaluation Metrics

The objective of this simulation is to optimize traffic flow. To measure this, the following will be measured each hour of the day:

- Average time from each source -> destination.
- Average vehicle speed including traffic intersections.

- Average fuel consumption.
- For each intersection, of light statistics over course of the day versus “stock” logic.
- Timing of traffic bursts so we can see the impact

Project Design

Reinforcement Learning

At its core, this will be approached as a reinforcement learning problem. To simplify things, this will be treated as a discrete space rather than continuous as modeling the physics of the vehicles is not needed here, and the action space is finite. This would not scale well for a “real” traffic simulation, however.

Discrete Event Simulation in the Environment

The environment itself is as complex as the agent and will be designed to:

- Generate traffic based upon the frequencies and source/destinations describes earlier,
- Keep track of each car as it travels from source to destination,
- Produce “video” data to feed into the agent,
- Evaluate and reward the agent based upon its choices.

The Main Loop

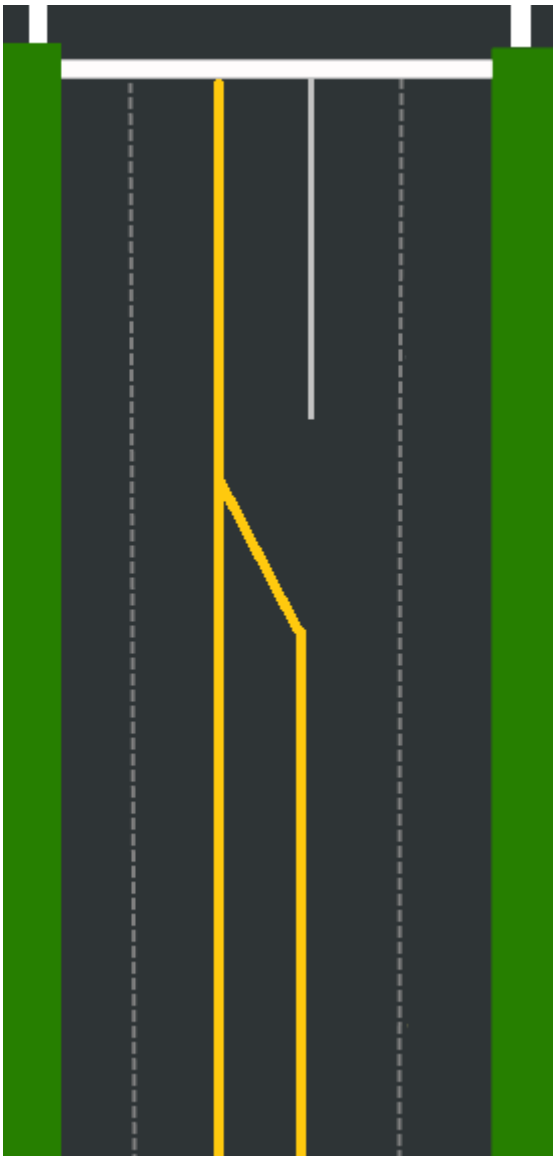
The primary loop will be driven by time, with a timestep of 1 second. The interval must be relatively short because the traffic cycle occurs in a degree of seconds.

```
Day = Monday
While not yet done with Friday
    Time = 8am
    TimeStep = 1 second

    While not yet past 8pm
        Agent.act(…)
        task.step(…)
        agent.step(…)
        state = next_state

        Time = Time + TimeStep
```

Video Sensor Generation

The video feed will be updated once per 15 seconds and will be available to the agent as a “snapshot”. Therefore, the snapshot could be out of data by as much as 15 seconds. The range of visibility will be about ¼ mile – cars can be detected approaching from that distance or closer.

For this simulation the video will be an overhead view rather than perspective. This prevents the need for complex parallax computations and questions of how to handle object occlusion. Every simulated 15 seconds within the timestep, cars will be place upon the following empty image based upon current position and direction of vehicles. This is cropped – as stated, the actual size will allow for ¼ mile visibility:



Traffic Generation and Tracking

For each timestep the Environment will generate new traffic based upon the expected frequencies and source/destination pairs. Every car will be tracked individually to keep track of where they are in their travels. Once they reach their destination, the vehicle is released and no longer tacked apart from any statistics that are accumulated from it.

Each vehicle will be represented with an instance of a class looking something like this:

```
Class Vehicle
{
    Int StartingPointID
    Int DestinationPointID
    Point Position
    Double Velocity
    Double TotalFuelConsumptionGallons
    Int TimeStopped
}
```

At each time step the cars will “move” and their states will updates accordingly.

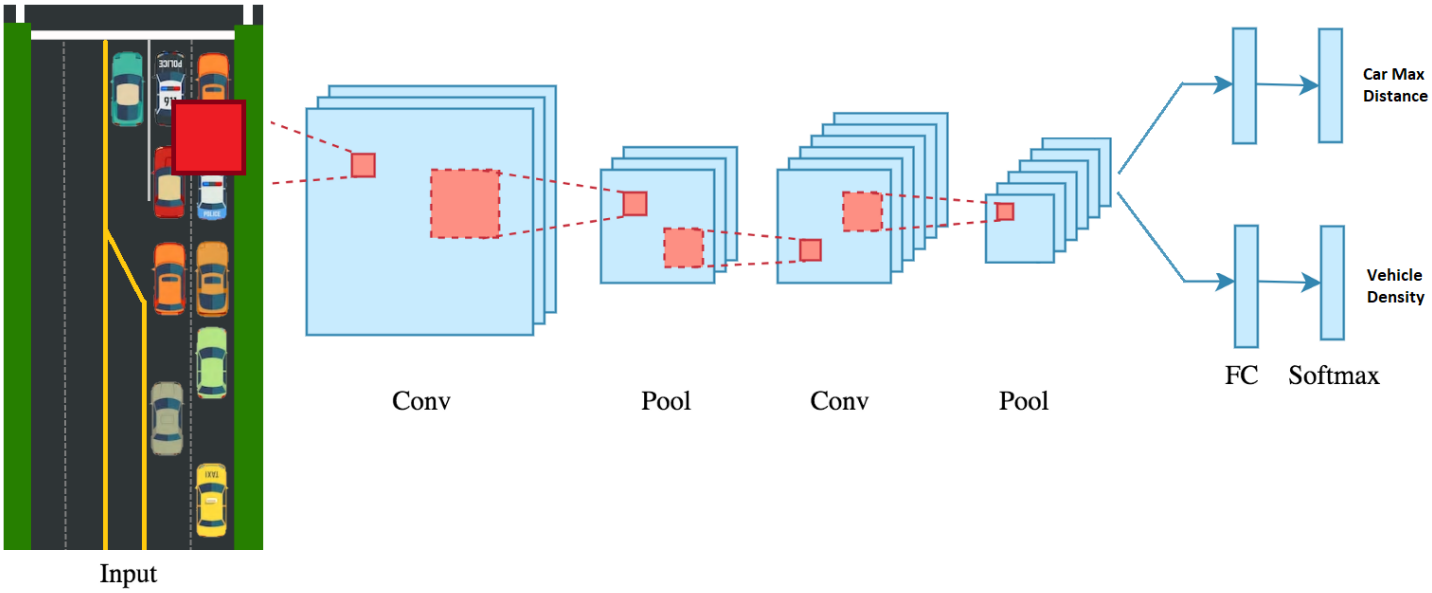
Video Snapshot Processing / CNN

The purpose of the video signal is to allow the agent to anticipate vehicles before they have yet reached the agent’s physical stop line sensors. The agent wants to know:

- Whether cars are coming,
- How many cars are coming, and
- How long until they will arrive

To determine this, the agent will assess the video feed at every step where all lights are in a red state. It will then use the data to adjust its timing, and this will form the basis of the experiment.

The agent will process the image data through a CNN that is trained against artificially generated data which is created against the described background.



Output from the CNN will be used as input to a MLP that will compute light transition times.

Technology

The project will be implemented in C# using a .NET wrapper over TensorFlow. C# was chosen because:

- It will allow me to create real-time visualizations easily for debugging,
- It is a complex simulation and C# is my “native” language and I want to focus on the logic itself.

The simulation will be using the GPU-based version of TensorFlow, running on a local Nvidia GTX960 CUDA GPU.