Machine Learning Nanodegree

Capstone Proposal

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.

Problem Statement

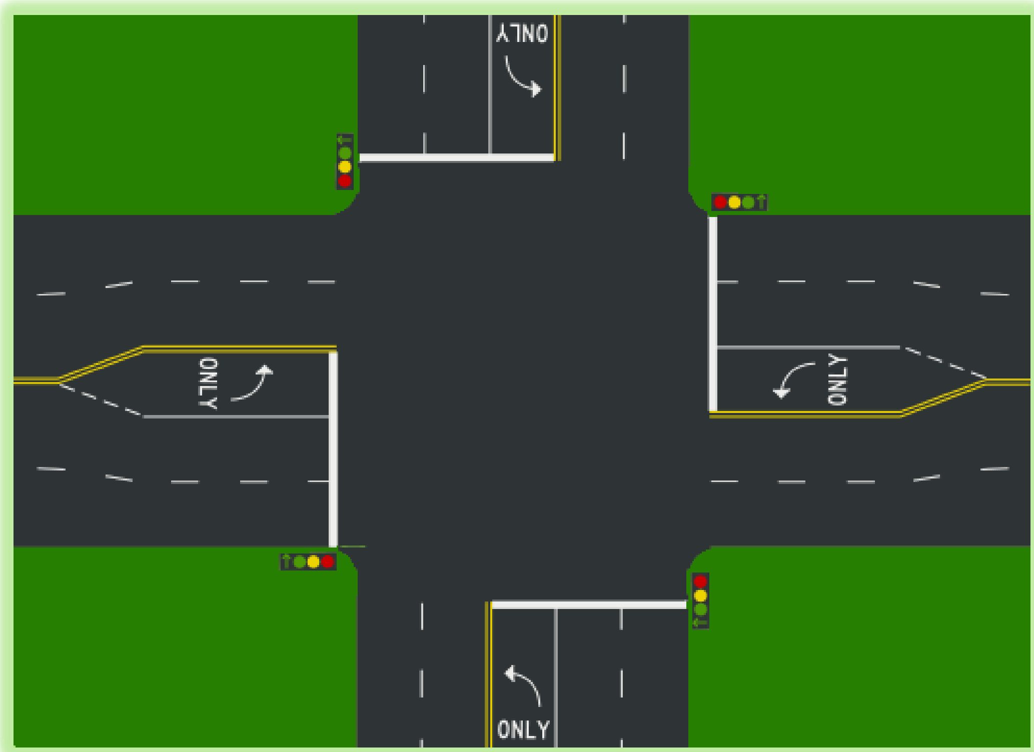
What We are Trying to Achieve

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

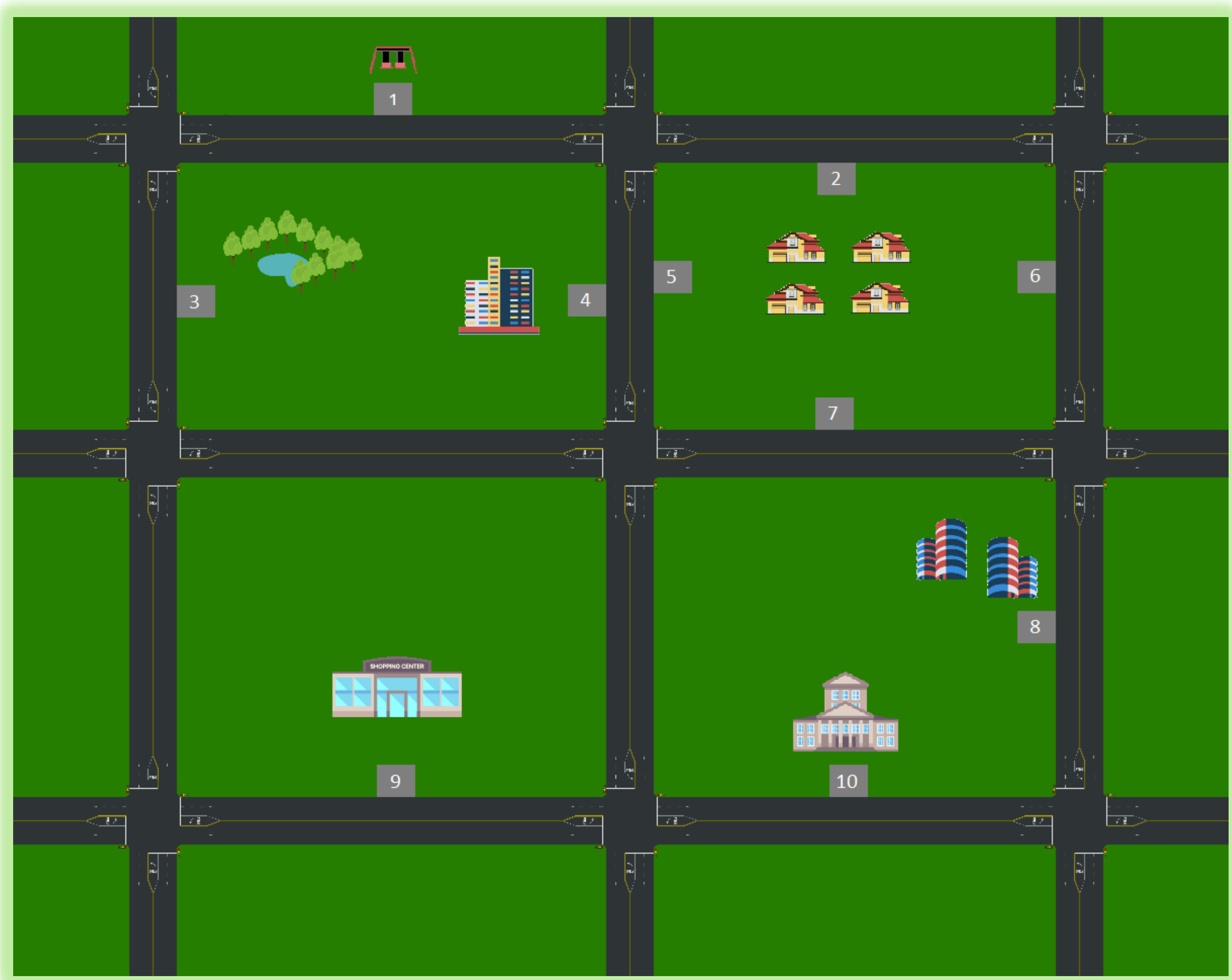
Intersection Layout

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

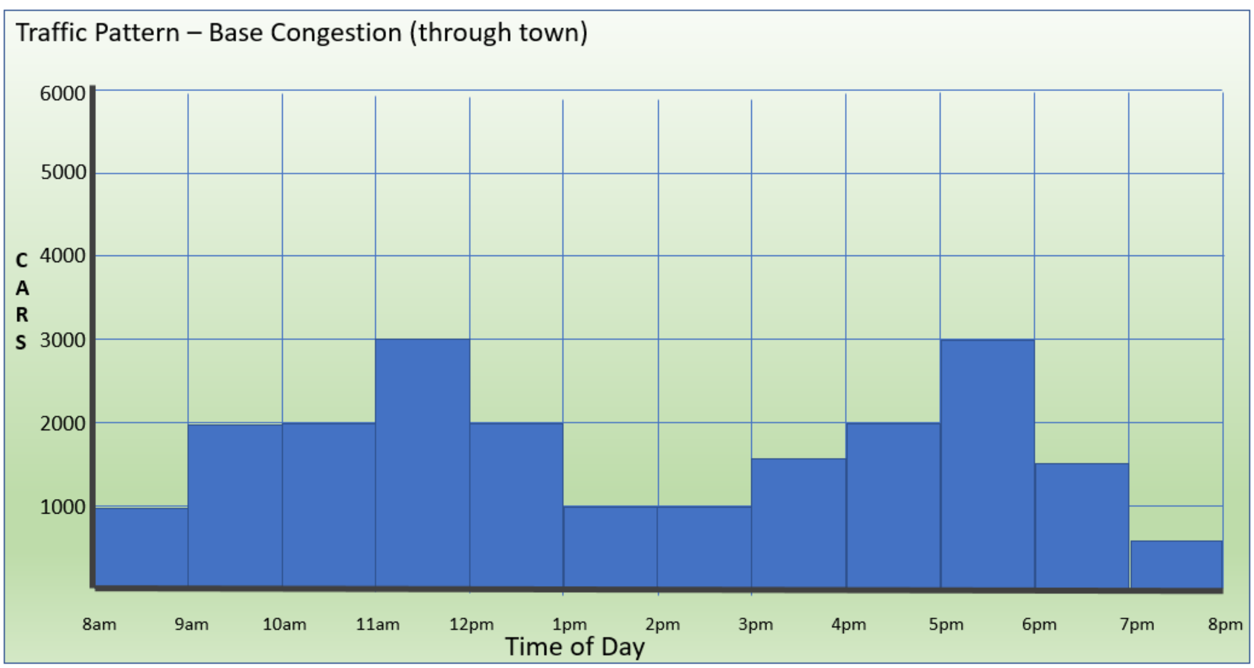
This problem is being defined as a reinforcement learning problem combined using discrete event simulation. There are two models that must be maintained for this complex simulation – one for the agent, and one for the environment.

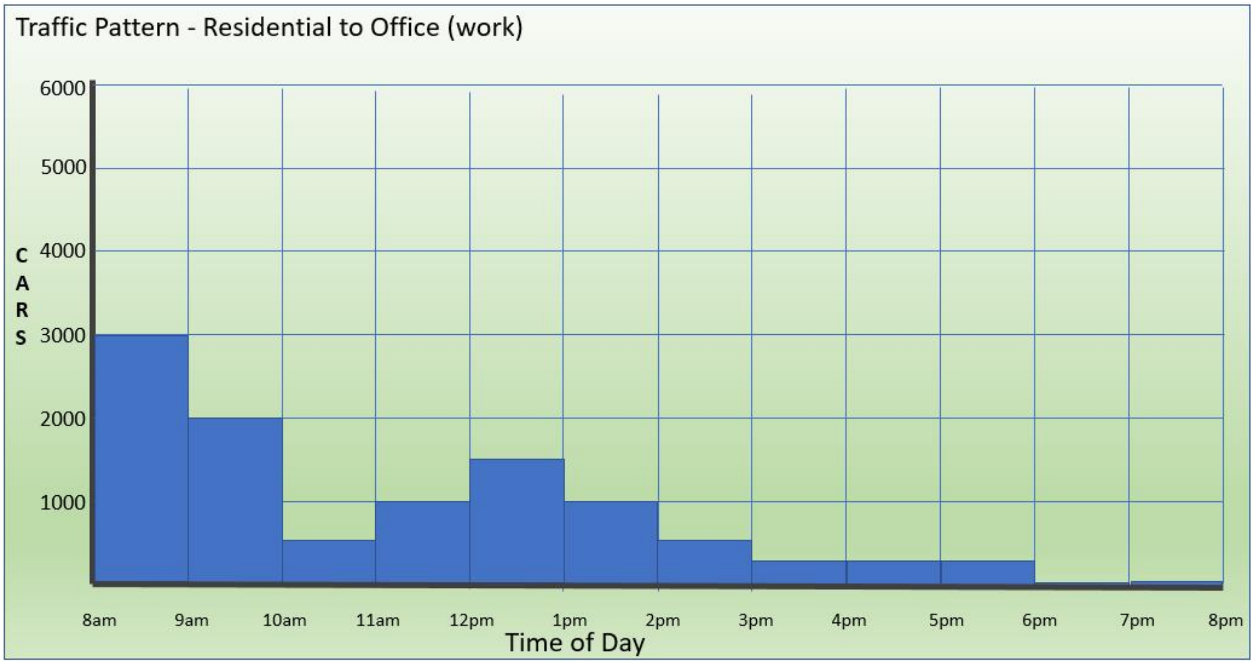
The environment must model:

- 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

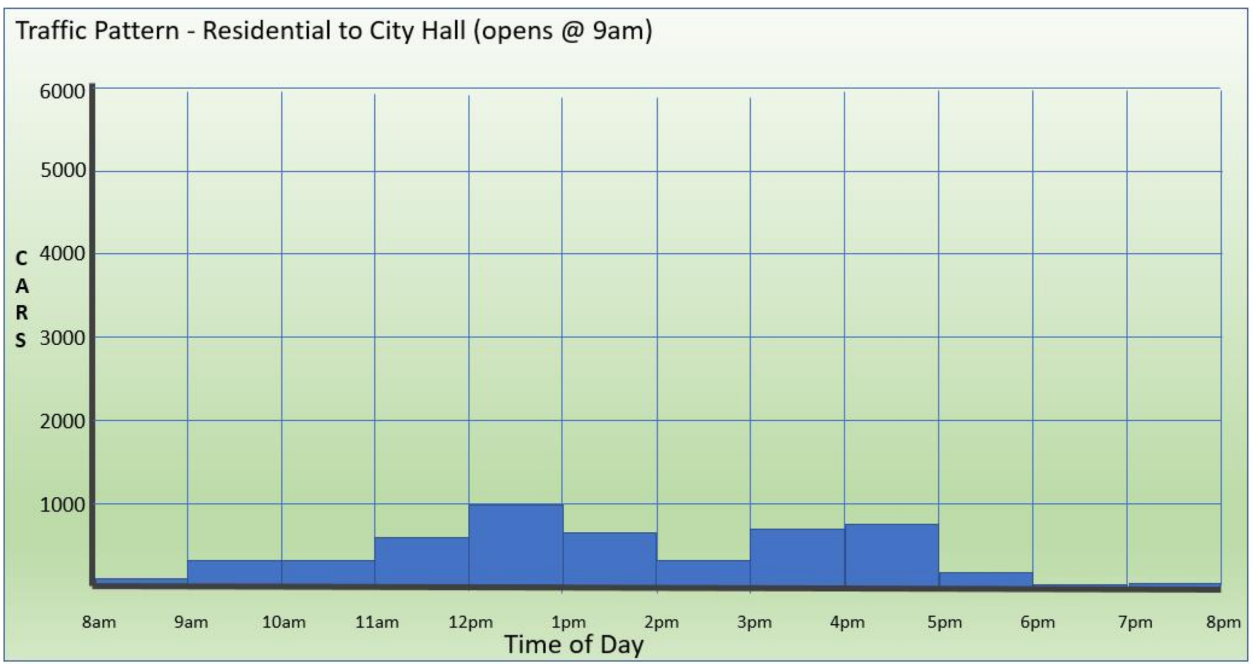




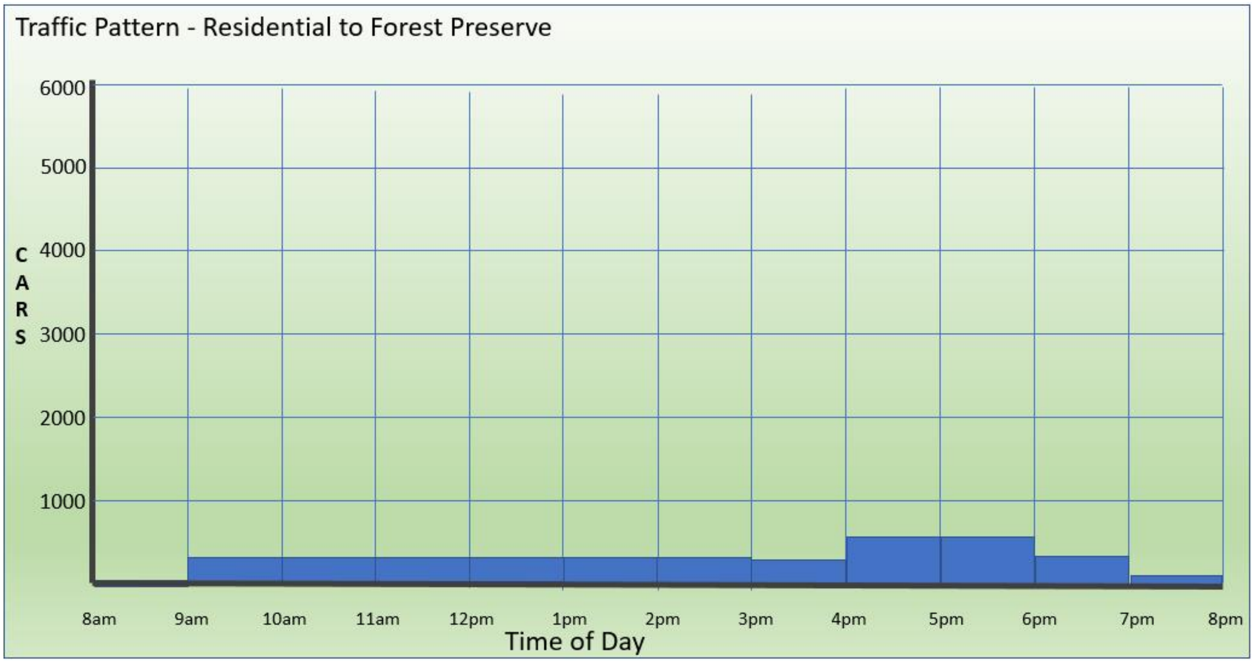
Traffic Pattern - Residential to Office (work)
6000
5000
4000
3000
2000
1000
CARS
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day

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)
CARS
6000
5000
4000
3000
2000
1000
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day



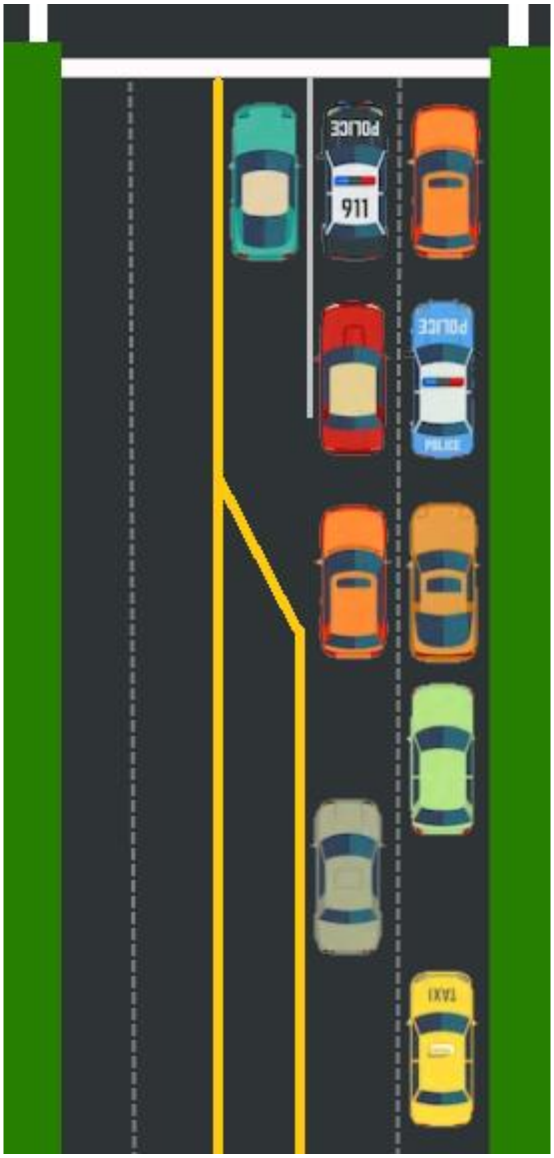
Traffic Pattern - Residential to City Hall (opens @ 9am)
CARS
6000
5000
4000
3000
2000
1000
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
Time of Day



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

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

- 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.

Project Design

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 GPU.