Traffic Control and Infrastructure Organization Using Reinforcement Learning

Consultant: Dr. András Lőrincz

2022 Fall Semester

Abstract

The purpose of this research was to develop a reinforcement learning model that can optimize traffic flow in a city by building roads, lanes and intersections between predefined nodes that are defined on a user-friendly interface. The model will take into account how fast and dynamically does traffic flow and how friendly a city is to its residents: the number of lanes and infrastructure cost have to be minimal while also minimizing the time taken to travel between junctions.

To conduct the experiment, the machine learning model uses a simulated environment that incorporates an intelligent driver model in order to provide the most accurate mapping between real life and the modeled environment. Future use cases include optimizing current road structure and generating new paths between already existing cities or settlements that use the least amount of taxpayer money while also providing them with the fastest travel times. The developed framework can also be generalized for uses in public transport: using smart forecasting to optimize bus or train lanes to cover the most area, for as many people as possible.

Contents

Part I

Introduction

1 Historic overview

In recent years the traffic of cities became a rising topic with more and more city governments realizing that a motor-focused city design is unsustainable. Cities all around western Europe have started designing their cities around humans and public transit, and not around cars. E.g. Paris is planning to be a 15-minute city, Barcelona is incorporating the superblock design, Finland is creating non-intersecting paths between the common intersections and the Dutch are using intelligent traffic light systems to manipulate traffic flow in order to optimize it for both cars and pedestrians.

If one takes a look at how the Dutch infrastructure is designed, they will see that despite having less car lanes and overall less space for cars, the traffic flows more smoothly. This is thanks to the intelligent design of intersections, traffic lights and infrastructure. The methodology of this has been known ever since the 1970s, but the auto industry has been fighting against it ever since in order to gain more market. In Northern America one can observe what happens if a city is designed with cars in mind, requiring everyone to own a vehicle in order to participate in society. This results in worse accessibility to the city for the disabled, incapable, elderly and young people as well. The methodology of how to create walkable, human-centered and intelligent infrastructure that is optimal for both pedestrians and cars is laid out in detail by books from authors such as Strong Towns, an urban planning organization.

The aim for this research is to able to model a system of roads or city, and being able to pinpoint mistakes made by development engineers, with the goal in mind to make the city more humanly livable, liquidate urban highways and make traffic infrastructure more efficient.

2 Goals and Outline

The project will focus on building an interface that models traffic flow in a graph-based structure, then train a reinforcement learning algorithm to find the optimal configuration of the roads in order to transport the most cars in the most effective way possible. Here's when the urban design principles come in: one can easily observe that the most effective way to transport as many cars as possible is if all roads are 8-lane highways. However it's also easy to see that it's miserable to live

in a city where there are no quiet, auto-low streets and only 8-lane highways. This might be the best configuration for cars, but it would make the life of people living in the city absolutely horrible. The rewarding system of the reinforcement learning environment will be designed in order to reflect these principles: building cost, traffic light/roundabout tradeoffs, how humans would feel living next to the road. The agent will have to decide where to build, destruct, or make roads 1-way to make the city's transportation flow dynamic but also make it livable for humans. The rewarding scheme will reflect the principles laid down by Strong Towns and other urban planning organizations significant in the field like Happy Cities: Transforming Our Lives through urban design.

The main metrics that are taken into account during the research are:

1. Time taken to arrive to the goal junction
2. Cost of infrastructure
3. Livability by humans (number of lanes, connection density)

3 Research hypotheses

If the research is successful, the following hypotheses can be verified or nullified:

1. Can traffic flow and driver behavior be modeled inside a reinforcement learning environment accurately enough for it to be representative of the real world?
2. Can predefined road configurations be optimized for traffic flow, human livability and cost efficiency at the same time?
3. Is there a single neural network architecture that can achieve the optimization in every configuration or there is a need for a more complex neural network as the complexity of the graph increases?

Part II

Research Background

4 City Design

4.1 A brief history of the modern day street

The shape and fabric of cities is the result of hundreds or thousands of years of development, transformation and reshaping. Many geographical and historical events play a role in determining how a city's transportation infrastructure is designed. The industrial revolutions have been de most markant inductors of change in most countries as they have led to the transformation of the walking-centered traffic. The most significant factor in city design has always been some idea of happiness or philosophy. E.g. the agora in ancient Athens, which was essentially a main square. However the philosophy of good life was built into it as it can be observed through monuments, temples, the most important buildings and law courts that have been surrounding it. And this practice of building and designing architecture hasn't changed ever since [9]. Even today, when a new skyscraper is built, it will reflect the architect's, CEO's or other executive members' ideas of happiness. They will look at it, and decide if they like it: is it good to go for building or not. But what has changed since ancient greece? What was the key motive that made us design cities focused around cars?

Philosophy has always been a pendulum oscillating between sky and earth - God and human. In the Middle ages, the focus was God. Then following it in the Renessaince, the human became the center. After that came the Baroque, again with God as the main motivator. After the enlightenment and the birth of the modern day citizen, the first style trend was Rococo, focused around humans. Around this time has religion lost its status as the main, unified cultural narrative as it can be observed in the works of Nietzsche and Derrida. The last attempt at constituting God as the center of philosophy was in the style trend of Romantics. However as the citizen of the enlightenment wasn't religious anymore, the style attempted to find God in the human (übermensch - Nietzsche). This is expressed in various other fields, not just architecture or city design: art, poetry, statuary, astronomy and even mathematics.

Around a hundred years ago in the 1920s cars have begun to appear in the cities in greater and greater numbers. At first it was a new and expensive technology, hence it was mostly the wealthy class that owned them. At this point in time traffic rules were essentially nonexistent, so drivers could drive however they pleased. This

quickly led to a large number of traffic related incidents. This was a massively bad PR for the auto industry, so they started influencing city design using their political and economical capital [16]. In school the children started to receive education about having to look both ways before crossing the road [19], because they are the ones that don't belong on the road and not the other way around. According to Peter Norton [18], the key turning point came in 1923, when the citizens of Cincinatti had to vote on a referendum that was planning to limit the maximum driving speed to 25 mph. The passing of this bill would have meant that car dealerships would have a very difficult jobs in selling their cars and essentially experience a drop in revenue, business and political influence. The representatives of the car dealers have did everything in their power to stop the referendum like billboards, flyers, public advertisement and even paying people to vote *no* on the ballot. Eventually the voting has turned in favor of the auto industry, and they gained political and economical influence over the course of the years.

4.2 The current state of city design

Ever since, car dependent infrastructure and lifestyle is overall in the USA. Owning a private vehicle has become a minimum requirement to take part in society as the sprawling has led to a run-down public transit and railway ifrastructure. As of the current state, the automobile is dominating the transport in the USA and hence becoming one of the most important determinants of the American lifestyle[21]. This can be attributed to the way cities and residential buildings are designed. A typical American city has overwhelmingly two types of residential buildings: skyscrapers and high-rise buildings downtown and low-density suburban sprawl around it. What's missing is a tradeoff between the two: mid-density, mixed-use housing that can at the same time house a sufficient amount of people in a relatively small area, but at the same time is relatively cheap to build in comparison with skyscrapers. This mid-density residential housing planning would be the key guarantee to design walkable neighborhoods in the United States, however it's missing. Of course, there are some areas where this type of architecture can be found but this is not the norm, it's the exception. This is referred to as the missing middle problem [20]. This is party because zoning laws in the USA don't make it possible to build this type of building.

From the way the cities are built comes car dependency as a consequence. However this has also led to public transit being under-developed: the housing density in the suburbs doesn't reach the threshold to make it worth making it being part of a bus or train route. No matter where the city puts the bus stop, the time taken to get there on foot in the area is too much for people not living near it. Putting bus

stops overall would require too much public transit infrastructure that would have to be subsidized from taxpayer money and would result in near-empty buses. The optimal scenario is when a transit route can reach just enough people so that the maintenance and operation can be financed from ticket fares and minimal amount of subsidies. This is called the critical density. If there is a critical density reached and a single transit stop can be serve enough people, which will make it worth in fare revenue. It's also rare to find dedicated bus and cycling lanes in the USA. Buses are generally stuck in traffic with the cars in rush hour, which makes it even less inviting for passengers. It's worthwhile to note here that adding lanes to an already existing road does not solve traffic, so it's not a solution to just make roads wide enough so that buses don't have to wait in line together with motor vehicles. As an example consider the recent expansion to the I-10 highway south of Houston known as the Katy Freeway. Together with a recent expansion, it's now at 26 lanes at its widest part but traffic jams are a regular occurence. Adding lanes also reduces the safety of the road, as found by [17] who were analyzing the relationship between the road geometry and safety, and also [1], who found a positive correlation in the numbers of crashes and the number of lanes in urban highway road intersections. Dedicated bike lanes are also missing from the infrastructure. The only bike lanes one can find are painted bicycle gutters without any separation from high-speed motor traffic that makes it very dangerous to ride a bicycle. This results in even more unequality in society.

Apart from the reasons mentioned before, car dependency has another different aspect, CO_2 emissions. A study has found that despite being 5% of the total population, the United States is responsible for 45% of all transportation CO_2 exhalation [5]. A different work in the field suggests that the most transport emissions could be reduced by converting low-density housing to mid-density [6]. Even more than converting mid-density to high-density.

4.3 The stroad

The stroad is a term created by a non-profit urban planning organization called Strong Towns that has been referred to and processed by several relevant pieces of literature like the book Confessions of a Recovering Engineer - Transportation for a Strong Town [15]. The idea of the stroad lays down the guidelines which have to be reflected in training the agent. To be able to understand the the stroad as a concept, it's necessary to first grasp the functions and purpose of a road and a street.

The road

The road is a high speed connection between places. Because of the high speeds involved the roads are wide and forgiving. The exits on the road are far apart to minimize the number of intersections and to keep vehicles at a high speed for the maximal amount of time. The more exits a road has, the more drivers need to pay close attention as the vehicle in front of them might be coming to a brake more often. This yields an extra danger factor to the driving experience. The solution to this problem is that on many roads there are separate turning lanes, which are dedicated to slowing down before taking an exit from the road. The lane next to the road is called the clear zone. It's a lane devoted to be left empty in case there's an emergency and a vehicle needs to stop or if an emergency vehicle needs to bypass a line of traffic. The signs the font on them are large, as they are meant to be read from a distance. An example for the road is the M3 highway in Hungary. Roads are very important pieces of infrastructure as they are optimized for safety while going fast.

The street

The street is a high-density environment that is designed for humans. It's a complex space where life in the city happens, so you will find people here. The street is dedicated to low speeds as it's a destination, not a through fare. Fast traffic speed is not compatible with human activity as it adds another danger factor to commuting. Buildings are right next to the sidewalk to be easily accessible by pedestrians. The feel of the street resembles something of an outdoor room. Everything is reduced to the scale of humans and it's designed to be encouraging and inviting for people to walk, socialize window shop or just simply get some fresh air. There are many entrances and exits to and from the street for shops, restaurants, caffés and services. Having an apartment near a street can also add value to the property. One can compare the price of a flat near Váci street and M3 highway and immediately see how much society values human factors taken into account when designing our city. Streets are very important factors for tourism, human livability and quality of life inside the city.

The stroad

The stroad is a name that's the result of the combination of the words street and road. It describes an urban motorway that is neither a street nor a road, despite trying to be both of them. The stroad is a street that's designed like a road, and doing so it fails to bring forth the advantages of either of them, but is struggling with the disadvantages of both of them. First the stroad is analyzed from the viewpoint

of a road.

On the stroad there are many entrances and exits to businesses and services, like one would find on a street. However there are multiple lanes of high-speed motor traffic in both directions. This is an added factor of danger, as there are typically no dedicated turning lanes, so drivers have to pay constant attention if the car in front of them is taking a turn. Because of the high speeds involved this means that drivers have to come to a quick halt. It's not rare that stroads allow the 70 km/h speed limit, which drivers often surpass, because the stroad is straight and wide. The geometry of the street allows the drivers to reach higher speeds while still feeling safe, however this is a fake sense of safety, as there are more points of conflict than on a dedicated highway [13]. Roads can keep vehicles on high speeds because there are few exits. Traffic jams are also a common occurence. Because of the large distances incolved, walking feels very uncomfortable in this kind of environment. People don't feel safe walking next to high-speed motor traffic. It's also common that one can find large parking lots, which are also not inviting for the human as there's nothing to see or do there. It's also difficult to cross the stroad unless using a dedicated crosswalk and treading very carefully not to be involved in an accident. This is clearly a place for cars so it's not surprising that pretty much everyone drives here.

Next the stroad is evaluated from the point of view of the road. On the stroad the lanes are wide and there are a lot of them. Traffic flow is more chaotic than on the road because there are a lot of exits, hence cars change lanes more often, easily giving the change to dynomen situations: when there's no imminent danger, but the conditions can turn dangerous at any moment. This also reduces the speed of traffic flow. Cars changing lanes are the leading cause of the caterpillar or butterfly effect. Traffic volumes are also high as often there are no viable alternatives to driving: bus and cycle lanes are nonexistent. This kind of car dependent infrastructure creates distances that only cars can bridge, but not everyone owns a car in society. This city design inherently widens the gap between the different layers of society.

Stroads are also very expensive, as they are built to a highway standard despite the fact that they can't be used as efficiently: with high speeds, while maintaining a strict level of safety as in the degree of slope, sharpness of turns and several other design decisions that have dedicated highway engineering and maintaining offices in almost any country. The density of intersections requires the city to install traffic lights, as they are the only type of intersection that can be operated with a related safety for a road of this scale. Traffic lights in the US can cost up to $750.000 for an intersection. Roundabouts and right hand priority intersection are not viable options here because of the high speeds involved. Drivers need to see if they have to slow down ahead of time.

The majority of crashes within the United States happen on stroads. People walking and cycling are especially in danger as there's no infrastructure that separates them from high-speed motor traffic. The wide and open design of the stroad encourages drivers to drive fast and not pay attention to their surroundings [4]. As an example, during the coronavirus lockdowns in the beginning of 2020, people weren't travelling as much, hence miles travelled by drivers decreased. Despite 355 million less miles driven fatal car crashes per mile increased by up to 34% [2]. This leads to the conclusion that the bad design of the stroad isn't killing even more people is that it's because they are usually so jammed up that the drivers can't go fast enough to kill eachother.

5 Practical uses of the research

Today a great percentage of a city's budget goes towards motor infrastructure planning, building, expanding and maintenance. Many cities mostly in North America keep building more new infrastructure that they can maintain causing a municipal insolvency. This is referred to as the "growth Ponzi scheme" named after the common investment fraud strategy in the literature [14]. It stems from the fact that maintenance costs start to appear after decades of road use, where it will be constant. It's easier to build roads than to keep fixing all the roads that are in bad condition. It also has political motivations as building a new road can serve as better PR than fixing potholes. There's also a large rate of suburban sprawl in the US and Canada, and lower density results in more infrastructure being built. This is an urban design and an infrastructure design problem at the same time.

There are many solutions from the urban design side such as building more dense housing or replanning highway expansions to not destroy historically dense neighborhoods that serve a good amount of people with affordable housing and public transportation. It's also very difficult to run transit lines through the suburbs because of the low density. There's a lesser amount of people in the rational radius of the bus stop where it's worth taking the bus rather than to walk a long distance to a bus stop and then take the bus. For this reason it's not common for US and Canadian suburbs to have widely accessible transit lines going through them.

The other principle is infrastructure design, which is the main focus of this research. From an infrastructure design perspective careful preparation and planning must precede the execution of building a new piece of infrastructure or the expansion of an existing one. The city must thoroughly consider where to spend its finite funds. This is the aspect where this research can help in the decision making process. The developed software and framework considers an exisiting piece of infrastructure between junctions and optimizes it in terms of traffic flow and human-centered

perspectives. The goal of the deep learning model is to find the optimal configuration which costs the least amount of money while providing a maximal amount of capacity expansion while still considering human factors in a traffic environment.

There can be three very distinct use cases of the piece of software that's being developed under the umbrella of the research:

1. Within-city: this use case comes down to optimizing an existing piece of infrastructure. This use case has some restrictions:
   (a) No new roads can be built between junctions as this would necessarily mean destroying existing buildings.
   (b) Only a limited number of lanes can be added to the existing roads.
   (c) Junction types can be set freely.
2. Green field investment (Outside-city): in this problem the RL agent is given junctions and it's free to set everything between them. The hypothesis is that there are no existing roads, and the junctions must be connected to each other in the best way possible:
   (a) Lanes and roads can be added freely.
   (b) Maximum number of lanes can be anything (within rational limits).
   (c) Junction types can be set freely.
3. Reparation mode (mixed): This is a mix of the previous two use cases. The RL agent is given a current traffic system and the task is to convert it to the best possible model while using a limited amount of money and resources.
   (a) The junctions are given.
   (b) Lanes and roads can be added, existing ones can be destroyed.
   (c) Junction types can be set freely.

If the human factors weren't taken into consideration the problem could be reduced into an optimization scheme where the target is to find the fastest traffic flow. The solution to this problem seems rather trivial: building 8-lane freeways between all the junctions. However a city like this can't be imagined within rational frames. The city budget and the experience of pedestrians must not be left out of the equation.

6 Deep learning in Infrastructure organization

6.1 The spectrum of expert areas

Predictions in new literature all point in one direction regarding traffic control and artificial intelligence: AI will most likely become the brain of infrastructure and traffic. With the rise of autonomous vehicles this possibility seems closer than ever. There's a huge potential in operating, managing and monitoring transportation. The infrastructure is going to be very similar to that of the internet currently: a decentralized network that is operated by mesh and local servers, data flow being sensor readings coming to and going from the several vehicles on the road. There are currently existing studies on the viability of smart networks that infuse different kind of sensors with road infrastructure [8]. For public transportation the LOA and GOA frameworks lay down the principles of creating an ethical AI-driven solution for trams and trains respectively [23]. There are also works that use traffic light governance systems enhanced by artificial intelligence to aid faster traffic flow [24]. Reinforcement learning has also been applied to control the cycle, phase and red-green ratio of traffic light systems [28]. All of the studies in the field can be placed on a spectrum that has the vehicle on one end and the road infrastructure on the other. Some works develop systems that enhance the vehicle, some are creating instruments for the road that directly influence traffic safety or speed like traffic lights. Others connect the road and the car together by infusing them with intelligent sensors that add a level of safety to the travelling experience.

6.2 Infrastructure and building modeling

Building information modeling (*BIM*) is the collective name for technologies and methods that aid the generation and modeling the digital representations of places in order to have a better overview of the building process. This introduces cost effectiveness, a level of safety and better ways to understand environmental impact. Today transportation infrastructure is a keystone of its economy by providing fast travel times for goods and orders, providing jobs thereby significantly helping economical and social development. In recent years a great need was shown for technologies that can increase the effectiveness of building motor transportation infrastructure. There was a significant development in the planning process for infrastructure. One of the results of this development is BIM. BIMs are essentially computer files that contain information on an object in order to help decision makers and planners with design, building and overview. Different kinds of dynamic data can also support BIMs like sensor measurements and signals from the building systems. Advances in fields like computer vision and artificial intelligence has given

rise to a technological revolution in building information modeling. There have been attempts where building models were constructed using a combination of LIDAR and photometric sensor data. BIM data itself is not an AI-driven process but it can help aid AI-driven dynamic modeling by providing more accurate information on buildings and infrastructure that can be part of a data-driven solution's environment model. BIM can be seen as both a 3D model and the information that it represents. In this case the model is more important as the representative of the information, not just solely a model file [10].

BIM by itself is not a very new technology. In fact it has been around since the 1980's. However combined with modern AI systems it can provide us with more accurate information and in depth-modeling. However in the recent years the interest in construction tech has grown significantly. Construction tech is a larger collective term for technologies and methods that aim at automating construction and building processes. These can be mostly categorized into three categories [22]:

1. Planning: applications that aid the planning phase. These are mostly software solutions for the designing and simulation of an environment. Some companies are experimenting with AR (augmented reality) where the planned building and the current state of construction are merged into one simulated reality.

2. Sampling: these are software and hardware solutions that have the goal of extracting information from the site. They are mostly sampling and measuring devices that are now becoming far more advanced than a simple laser ruler. The LIDAR building scanners are an example. The output of these devices often get fed into the software that's responsible for planning and modeling.

3. Construction: robotic hardware tools that actively build or destroy infrastructure on the job site. Big leaps have been made in transportation of material, applying paint and cleaning surfaces.

In the US some estimates say that the amount of venture capital invested in construction technologies got as high as $1000 million in 2018. This is a 400% increase from 2013 [3]. This shows a large peak of interest in such technologies and gives reason for further research in such fields of expertise.

Part III

Methodology

7 Constructing a city

Everything starts with a city that the agent will have to optimize the structure of. At this point the city is best described by a graph. Graphs are high-level descriptors of connections between nodes. This gives an easy to use but highly customizable method to define a city to optimize. To define a city using the graph there are two things needed to be defined:

1. List of nodes of the graph with coordinates: $N_1, N_2, \ldots, N_k$.
2. List of connections between the nodes: $N_i \to N_j,\ i \in 1 \ldots n,\ j \in 1 \ldots n$.

Together they are called a *construction protocol* from here on.

The developed software product will provide an interface where the user can make a graph, describing the intersections (nodes) and roads (edges) of the city in question. This for example can be done in GeoGebra and exported into a construction protocol in order to work as an input for the model. Below is an example simple city constructed:

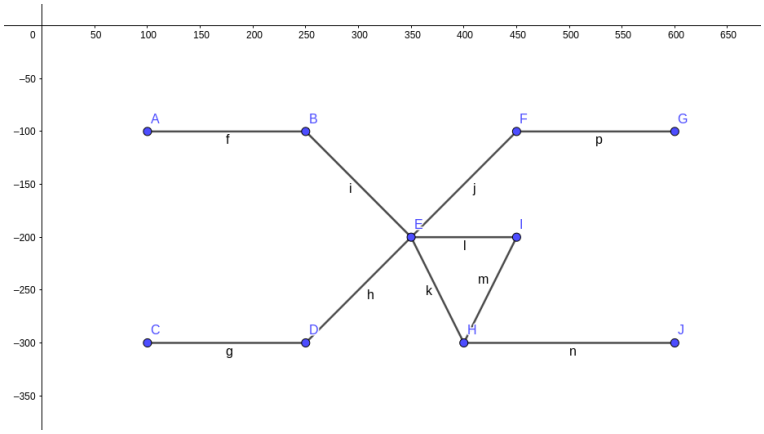

Figure 1: Graph-based representation of a city

The interface will read a construction protocol, construct a graph and all the possible pathways from it. As a starting configuration every road is 2-way on the edges that have been defined in the graph. The vehicle rate and distribution can be controlled before starting the simulation. A constructed "starting" city according to the previously shown graph will look like as follows:

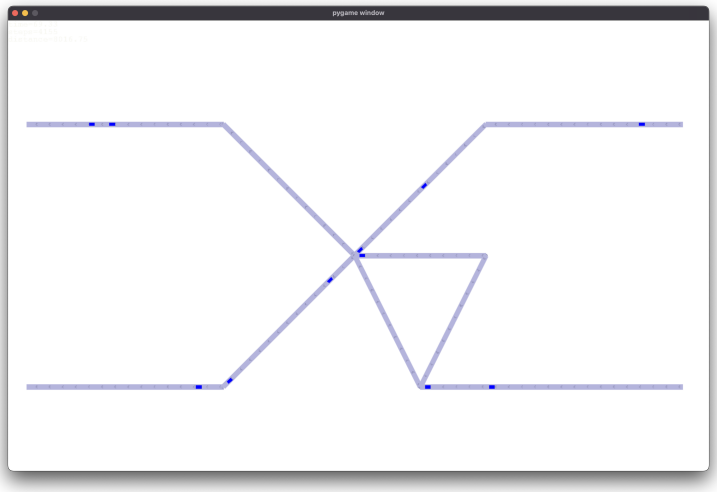

Figure 2: The simulation running the same city. The dark rectangles are cars on the road.

So far this is a simple setup for demonstration. The vehicles are passing from one entry point to another, without necessarily choosing the shortest path, or being evenly distributed among all the roads, just as one would find in real life. The driver model will incorporate an intelligent behavior, like slowing down after the car in front is slowing down or gradually speeding up after a light has turned green with a comfortable acceleration parameter.

The road configuration will be examined with multiple metrics like how many steps does it take for the roads to transport 100 cars or how much the road infrastructure would cost. If the agent is handed a road configuration it will be able to find the optimal one, with the least cost, least unnecessary roads and fastest transportation for a given amount of cars. The cars are generated using a predefined distribution and each traffic junction has a separate entity of vehicle generator attached.

8 Intelligent driver model

The intelligent driver model is a time-continuous car following model for the simulation of urban traffic. It describes the behavior of the drivers. and the positions of vehicles. The model incorporates the influence of one vehicle to another e.g. when some vehicle stops, the vehicle following it has to stop as well just like in real life. This property makes the model a good choice if we want to simulate real-world traffic as traffic jams, flash congestions and highway-like behavior is built into the model. The followin 6 parameters influence how a driver behaves:

1. v_0: The desired velocity of the vehicle
2. T: Safe following time
3. a: Maximum acceleration
4. b: Comfortable deceleration
5. δ: Acceleration exponent
6. s_0: Minimum distance between vehicles

The original model and the model that was used in this research was invented by Treiber, Hennecke and Helbing [26]. For the sake of simplicity the same parameters are used all over the simulation. The dynamics of the vehicle regarding acceleration and deceleration counting in the leading vehicle are then described by the following equations according to Treiber, Hennecke and Helbing:

$$\dot{x}_\alpha = \frac{\partial x_\alpha}{\partial t} = v_\alpha \tag{1}$$

$$\dot{v}_\alpha = a^{(\alpha)} \left[1 - \left(\frac{v_\alpha}{v_0^{(\alpha)}} \right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha} \right)^2 \right] \tag{2}$$

The deceleration depends on the ratio of the desired minimum gap and the actual gap:

$$s^*(v, \Delta v) = s_0^{(\alpha)} + s_1^\alpha \sqrt{\frac{v}{v_0^{(\alpha)}}} + T^\alpha v + \frac{v \Delta v}{2\sqrt{a^{(\alpha)} b^{(\alpha)}}} \tag{3}$$

Using these interpolations of the variables and the positions of other vehicles it is possible at every time step to completely determine the position of every vehicle. Furthermore, vehicles can influence each others' positions, as there can be car pileups in an intersection with cars waiting for each other. This model will consider only cars of uniform size, without taking into account other types of vehicles like vans and semi trucks. Every driver is assumed to have the same skill set e.g. safe following time, comfortable deceleration and so on. This can be a fairly good approximation of a real world driver's parameters.

The simulation of the drivers' behavior is highly dependent on time, as they are regulated by differential equations that are implemented using computer code. Each time the agent will execute a training iteration, it will have to run a simulation. However, the simulation is not possible to run so many times in real-time. The game-time is implemented using discrete time steps, that can be sped up to arbitrary speed to run the simulation as fast as possible.

The IDM is very important in terms of the reinforcement learning problem, as traffic jams can be measured by calculating how long each car idles, how much distance did they cover while taking into account how many cars were spawned in the simulation. Part of the agent's reward is based on this term. If the cars covered more distance the reward will be higher, whereas if the cars idled for more time or traffic jams happened, the reward will be lower.

9 Modes of operation

There can be several different ways to use the framework depending on what the target of the prediction is. The modes depend on what actions are allowed. The more actions are allowed, the more the city will be redesigned from the ground up. The mode of operation is in a very strong connection with the use case of the of the framework (within-city, green-field or mixed) and essentially defines the types of actions that are allowed for the agent for specific usages.

1. Lane capacities: in this operation mode the nodes and junction types are fixed. The agent is only trying to predict how many lanes should be going from one intersection to another. The agent can add and remove lanes to existing infrastructure. This is the ideal mode of operation if there is a currently existing city, with already existing streets as it's not feasible to demolish residential apartment blocks just to build a new road between two intersections.

2. Lane capacities and roads: this is the same as the mode of operation mentioned before, the only difference is that the agent now has the capability to add a new road between junctions. This is the optimal configuration if it's the focused map segment is not already inside a city, but rather in a rural area where fast connection to nodes is a matter of hours of travel time on the highway.

3. Lane capacities and junctions: in this case the agent is not allowed to build any infrastructure that would mean destroying a piece of the city. It's only allowed to take the already existing roads and add or remove lanes from them, but not to build new roads where there isn't already one. The type of junction can be switched between righthand, roundabout and traffic light as it's still feasible to execute these types of modifications without needing to demolish residential properties.

4. Junctions only: the junctions only modality focuses on the nodes of the graph exclusively. The goal here is to determine what intersection types would be best fit to control traffic flow in an already determined road configuration. The agent has to choose between right-handed, roundabout and traffic light

for every junction in the map. Junctions only is the optimal setting in an environment where there's a narrow city e.g. like in Western Europe but there's room for optimization in terms of intersections in order to determine the configuration that yields the fastest traffic flow.

5. Junctions and roads: this is the hardest task in terms of optimization and urban design. The mode allows the agent to set each intersection type, add lanes to already existing roads and add new roads between junctions. Newly built cities, already existing districts that are to be redesigned and between-city highways, country roads can also use this method of optimization. The research will focus on this mode as this is the most general.

10 Reinforcement learning

As this is a reinforcement learning based research so the following section will lay down the fundamental concepts of reinforcement learning. The definitions shown in this section are described by Sutton and Barto in their work Introduction to Reinforcement Learning [25].

10.1 Basics

Q-learning

The basic function for the model to estimate is the state-action value function, which is the expected value of the sum of the cumulated discounted future reward, given state s_t and action a_t:

$$Q_\pi(s,a) = E_\pi\left[G_t \mid S_t = s, A_t = a\right] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \quad (4)$$

Where γ is the discount factor, and R is the immediate reward received by the agent for its action. This expectation value is the one that the model will have to estimate. Before diving into that, first we will look at another approach, which is the Bellman optimality equation for the state-action value function. This is a very compact formula that uses the dynamics of the environment to calculate the $Q(s,a)$ value in a closed-form equation:

$$Q_\pi(s,a) = \sum_{s',r} p(s\prime, r \mid s,a)\left[r + \gamma V_\pi(s\prime)\right] \quad (5)$$

Where the term $p(s\prime, r|s,a)$ denotes the probability that the agent will receive reward r, end up in state $s\prime$, given that it's currently in state s and takes action a,

and thereafter following the existing policy, π. $V_\pi(s\prime)$ is the state-value function in the next state, $s\prime$. The issue with this formula is that despite it being very compact, it might take a lot of processing power to compute in complex environments, and also that the dynamics of the environment is not known for most cases (like in the case of traffic control). This is why it's reasonable to turn to deep learning methods to solve the task in order to estimate the $Q(s, a)$ value function. With the help of effective code and fast deep learning libraries, good accuracy and speed can be achieved.

Embeddings in deep learning

Embeddings are a common way to represent discrete categorical variables as continuous vectors. The most ccommon use case of embeddings is in natural language processing, where words are represented as dense, high-dimensional vectors. The main principle behind learning embeddings is that similar features should have similar embedding representations. It is possible to train the embeddings' weights using backpropagation in a neural network. It is also common practice to use embeddings as inputs to other layers after they have been learned. The key feature of embeddings is that they are defined by their similarities and relationships. This comes in handy when performing a natural language processing task as the model can learn how different words are related to eachother and thereby find context, meaning. Different words can have the same meaning despite having a very different character representation e.g. *'fantastic'* and *'wonderful'*. The neural network learns the similarity of these words by training their embeddings to become similar.

In this research the embeddings were used to represent the nodes of the graph. In graph neural networks embeddings are used to capture the structure and relationships within the graph, allowing the network to perform tasks such as link prediction and node classification. When a GNN learns an embedding it learns to map the nodes and edges into a continuous vector space such that similar nodes and edges have similar representations and dissimilar nodes have dissimilar mappings. The embeddings in this case are able to capture topology, node attributes and the global structure as well. For example, GNNs can learn to represent nodes based on their degree, their position in the graph, the attributes of their neighbors, and the overall connectivity of the graph.

Graph convolutional networks (GCN)

Graph convolution is another approach to extract structural and representational information from a graph. Graph convolutional networks work very similarly as their image-based counterparts except that the input data is graph-structured. During

a pass in a GCN layer, the information is not passed in a grid-like fashion as in traditional image-based convolution, rather information propagation works between neighboring nodes in the graph. GCNs can be used for node classification, link prediction, neighborhood detection, local and global pooling of nodes. The resulting information can be understood as embeddings of the nodes which capture the relationships and features of the nodes. Each node will have a custom embedding in the number of dimensions of the output layer's neurons.

The general propagation rule for a GCN is defined layer-wise where the input is a graph's adjacency matrix and the weights of the nodes in two separate data structures [11]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{6}$$

Where $\sigma(.)$ is the activation function, $\tilde{D_{ii}} = \sum_j \tilde{A}_{ij}$ is the degree-matrix of the graph, $\tilde{A} = A + I_N$ is the graph's adjacency matrix including self-connections, $W^{(l)}$ are the trainable weights of the layers and $H^{(l)}$ denotes the activations of the l-th layer in the network. The weights of the edges are passed as nonzero scalars in the correct entry in the adjacency matrix. This research uses GCN layers in order to extract information before dense fully-connected layers.

10.2 Deep Q-Learning

Generic DQN

The deep Q-learning is an extension to the original method of calculating the state-action value function explicitly, using a (deep) neural network. In tasks where the state is defined by an image of the gameplay screen, the go-to solution is a deep convolutional network that processes the raw pixel data and outputs the action to be taken by the agent. In the case of this research the environment passes the city's graph's adjecency matrix as the input to the Q_learning agent. In classical Q-learning the state-action values are updated iteratively by the following rule:

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} Q_t(s\prime,a\prime) - Q_t(s,a)\right) \tag{7}$$

Where α is the learning rate hyperparameter. and $r(s,a)$ is the immediate reward received for taking action a from state s. As stated before, this tabular approach is not possible as the environment has very large state space. However the Q-learning update rule can be directly implemented using a neural network. In this case instead of the Q-table, the model will be updating the weights of the neural network called parameters. The original temporal difference loss will be replaced by a loss function L:

$$L(\theta_t) = E_{(s,a)\sim D}\left[\left(y_t^Q - Q\left(s, a, \theta_t\right)\right)^2\right] \tag{8}$$

Where y_t^Q is the target value. The term D stands for an experience buffer which contains important data on the episode like state, action, next state, reward and dones (booleans to tell if the agent has terminated with either win or fail). The experience buffer is sampled uniformly each time the model updates its weights and then emptied out. This originally was the idea of Long-Ji Jin [12] and is now widely accepted as a method of improving the model's performance. The loss function is calculated based on the sample and gives the experience for the agent to learn upon. This function measures how good the model performed on a given training iteration (episode). The procedure corresponding to the traditional Q-table update is gradient descent on the parameters of the neural network. It takes the following form:

$$\theta_{t+1} \leftarrow \theta_t + \alpha E_{(s,a)\sim D}\left[\left(y_t^Q - Q\left(s, a, \theta_t\right)\right)\nabla_\theta Q\left(s, a, \theta_t\right)\right] \tag{9}$$

Where ∇_θ is the gradient of the loss function.

Double DQN

In classical Q-learning and DQN, the algorithm always uses the same maximum values to select an action and to evaluate that action. This is what is referred to as optimistic estimation and this is something that is better to be avoided. The algorithm for Double Q-learning was originally proposed by Hasselt [7]. In this case two estimators are used synchronously to retrieve the the predicted action value. The two estimators are referred to as target and local networks. When updating the state-action value function, one of the models is used to predict the best possible action, and the other one is used to evaluate the taken action. For this reason Double Q-learning has to store two Q-tables called Q^A and Q^B. The values from one network are copied over to the other one every defined couple of iterations. This results in lower variance and more robust models. The update rule is the same as in case of the DQN, but right now the model has to update the online network. However there's a slight change that has to be introduced as a result of having multiple estimators in the system. The target value in case of DDQN is:

$$Y_t^{DDQN} = r_{t+1} + \gamma Q\left(s_{t+1}, argmax_a Q\left(s_{t+1}, a, \theta_t\right), \theta^-\right) \tag{10}$$

Where θ^- is the set of parameters of the target network, the one thats weights get updated every couple of iterations (defined by the programmer). The local network's weights get updated the same way as before. The update ratio is usually denoted

by τ.

Dueling DQN

Before discussing the DQN viewpoint for dueling network architectures, it's important to define the advantage function. This measurement is one type of value function that's obtained by subtracting the state-value function from the state-action value function and represents how good is it to be in a state s, take action a and thereafter follow the given policy relative to the other actions that can be taken at that given state:

$$A(s,a) = Q\left(s,a\right) - V\left(s\right) \tag{11}$$

The dueling neural network architecture was first presented by Wang et. al. [27], which explicitly separates the inner calculation of the state-value function and the state-action value function. The network's inputs are the same as in the cases before, the observations of the agent, and the hidden layers can be customized specific to the task as needed. However the main difference is in the output. The network will get two heads (output layers): one of the size 1, for estimating the state-value function of the current state and another one that's the same size as the action space (7 in case of traffic control) for estimating how good is it to take each possible action. Then after these values are calculated, the network reassembles them with the following forward mapping in order to maximize the Q-value:

$$Q\left(s,a,\theta,\alpha,\beta\right) = V\left(s,\theta,\beta\right) + A\left(s,a,\theta,\alpha\right) - \max_{a' \in |A|} A\left(s,a,\theta,\alpha\right) \tag{12}$$

The main principle of the dueling network architecture can be seen on the following graph:

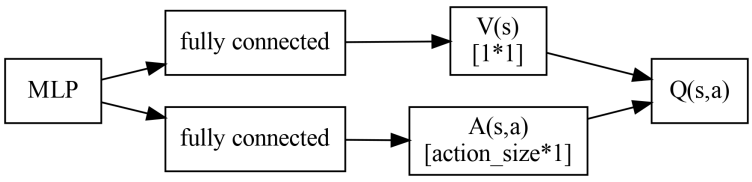

Figure 3: Dueling network architecture

The square node shape denotes deep learning layers with trainable weights.

Part IV

Deep learning architecture

11 The modeling procedure

The modeling process describes the methodology to train the agent. Training means finding the weights of the neural network. The algorithm was chosen to be a very standard reinforcement learning framework with very little additions.

The modeling structure is vastly dependent on the task that is given to it, therefore this section will only give a general overview of the inner workings of the deep learning model. The main steps of a single iteration of simulation are given below:

Algorithm 1 The modeling process

Input:

- A state vector x that describes connections between nodes in a graph G according to the current road configuration.
- Number of episodes $n\_episodes$.
- Number of steps in episode n_steps.

Output:

- The road configuration which is optimal for travel time, cost and livability.
- The deep learning models with trained weights.

Process:

1. For *episode* in $n\_episodes$:
   (a) $A \leftarrow \emptyset$
   (b) $possible\_nodes \leftarrow \{N_1, N_2, \ldots, N_k\}$
   (c) For *step* in n_steps:
      i. Agent observes state vector x.
      ii. Environment chooses *start* node $A \in possible\_nodes$ if $A = \emptyset$.
      iii. Agent chooses *end node* $B \in possible\_nodes$.
      iv. $possible\_nodes \leftarrow possible\_nodes \setminus A$.
      v. Agent chooses an *action* $a \in a_1, a_2, \ldots, a_n$ from the set of valid actions depending on the *state*, *start* and *end*.
      vi. The agent executes action a in the environment: $A \xrightarrow{a} B$.
         A. The environment builds or removes the piece of infrastructure.
         B. The simulation runs to a given stopping criterion (e.g. time elapsed, number of cars finished).
         C. The environment calculates and passes the agent the reward signal r and the modified state vector x.
      vii. The agent records *start, end, action, reward* and executes a learning step based on the reward r if needed.
      viii. $possible\_nodes \leftarrow \{N_1, N_2, \ldots, N_k\}$ if $possible\_nodes = \emptyset$:

It's easy to observe that the machine learning model has a particularly difficult job in this case. During the modeling process the agent has to step on the graph

thereby generating a path for itself. During each step it can choose the next node and what action to take on the next node. This is in order to encourage the agent for exploration and to make sure that it visits all the nodes in the graph. If the agent decides to build a lane an additional lane will be built from the current node to the next. If the agent changes teh junction type the junction type will be changed in the next node. Using the information extracted from the graph's structure around the starting node the agent predicts the ending node. Using the starting and ending node the agent will predict the action. The agent knows at erach time step which are the possible actions with respect to the current state e.g. it can't remove a lane between two nodes where there are no lanes in existence currently.

The action it has to take in each iteration consists of multiple variables, not just a single one: $A \xrightarrow{a} B$. For this reason a highly specialized deep learning configuration is needed. The input vector is the size of the state space as described by the node connections previously in this chapter The number and size of the hidden layers is highly dependent on the size of the input graph, so the rest of this section will deal with the problem of predicting B and a. First the *'start'* and *'end'* node prediction will be taken into account. For this there are two proposed architectures, the single and the multi-headed ones.

12 State space

12.1 Representation

The state of the environment can be described in all time instants with a directed graph where the nodes of the graph are junctions and the edges are the roads connecting each junction. The weight of the edges refer to how many lanes are currently built between two junctions.

First, the attention will focus on how to represent a certain type of road between two nodes. The state between graph node A and B will have to be represented by a single number in every case. The bases that this scalar value will have to cover the number of lanes between $A \rightarrow B$ and the type of intersection in a node.

Second, the representation of the intersections: they are technically arbitrary but the possible range of values was chosen as $[1, 2, 3]$ from the cheapest to the most expensive: $\{1 : righthand, 2 : roundabout, 3 : trafficlight\}$.

The state of the intersection is in its essence a graph, so the following descriptions will have to reflect this principle. All descriptions will be given in terms of graph connections. The representation of the graph is the topic of a following section. For a state-vector element corresponding to the one-way connection between nodes A and B the possible values are as follows:

1. One-way, one-lane road between nodes A and B: $A \to B \Rightarrow 1$
2. One-way, two-lane road between nodes A and B: $A \to B \Rightarrow 2$
3. Two-way road, one lane in each direction between nodes A and B: $A \to B \Rightarrow 1$; $B \to A \Rightarrow 1$
4. Two way road, two lanes in each direction between nodes A and B: $A \to B \Rightarrow 2$; $B \to A \Rightarrow 2$
5. Right-hand intersection in node A: $A \to A \Rightarrow 1$
6. Roundabout in node A: $A \to A \Rightarrow 2$
7. Trafficlight intersection in node A: $A \to A \Rightarrow 3$

The state vector keeps track of the connections in a directed fashion, storing the number of lanes from $A \to B$ and $B \to A$ in separate values. A self-connection always refers to the type of junction that is present at the graph node. From now on the weight of a connection between nodes A and B in the state representation will be referred to as $x(A, B)$ or $x(N_1, N_2)$.

12.2 Adjacency matrix based representation

A graph can be represented several different ways, each having its own advantages and disadvantages. For the reinforcement learning problem at hand the graph is represented using the graph's adjacency matrix. This opens up new possibilities and challenges as well. In the matrix-based approach each entry to the matrix describes a connection between two nodes, like a table. For n nodes the descriptor matrix x is of size $n * n$. The diagonal elements $x(N_i, N_i)$ describe the self-connections, so the type of intersection that can be found at a junction in the city. All other connections $x(N_i, N_j)$, $i \neq j$ represent the number of lanes between nodes i, j. The representation is as follows:

	N_1	N_2	$\dots$	N_k
N_1	$x(N_1, N_1)$	$x(N_1, N_2)$	$\dots$	$x(N_1, N_k)$
N_2	$x(N_2, N_1)$	$x(N_2, N_2)$	$\dots$	$x(N_2, N_k)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
N_k	$x(N_k, N_1)$	$x(N_k, N_2)$	$\dots$	$x(N_k, N_k)$

Table 1: Adjacency matrix representation

This approach yields advantages and disadvantages as well as the one-dimensional scheme. The downside is that the matrix's size is fixed: there cannot be any edges that can be left out of the learning. If they are fixed on the value 0 to denote this, there is still a necessity to process them at each training iteration as deep learning libraries rarely have an option to manually skip a connection. This would lead to complicated program code and would raise more errors rather than gain execution speed.

The advantage is that with this method there is now a possibility to use two-dimensional or graph convolutional processing as the matrix can be processed as a single-channel monochrome image.

Another key insight is that the order of the columns doesn't actually matter: they can be ordered in any order so long until the connection descriptor elements are also in the right place. This can be a useful trait to exploit as the nodes can be added in order of closeness to a keypoint or sorted by any other comparison metric. The matrix would now become not just arbitrarily ordered, but also the node positions would also have a meaning to convey.

13 Action space

Depending on the starting configuration the agent will have a given set of junctions and some to no roads to start with. From here on the goal is to add / destroy roads, lanes and change intersections in order to optimize the throughput, capacity and cost of the road. Each action of the agent will take two graph nodes as a parameter and the roads will be configured accordingly. Any action for node A and B will assume a single directed edge $A \rightarrow B$ starting from A and ending in B. There are cases where semantically it would make more sense to have more or less than 2 parameters but these cases can be generalized to an action with two parameters and hence be channeled into a neural network output of the same shape and size as all other cases.

The list of actions for the discrete action space:

1. $add\_lane(A, B)$*:* Adds a single one way lane going from $A \rightarrow B$.

2. $remove\_lane(A, B)$*:* Removes a single lane going from $A \rightarrow B$.

3. $add\_road(A, B)$*:* Adds two lanes between the nodes A and B going $A \rightarrow B$ and $B \rightarrow A$. Only valid in case of nodes that don't have edges connecting them.

4. $remove\_road(A, B)$*:* Removes two lanes between nodes A and B going from $A \rightarrow B$ and $B \rightarrow A$. Only valid between nodes that have edges connecting

them.

5. *add_righthand*(A, A)*:* Removes current traffic light or roundabout infrastructure to create a right-hand priority intersection in graph node A.

6. *add_roundabout*(A, A)*:* Adds a roundabout to all roads entering the intersection of graph node A. Destroys current trafficlight or right-hand intersection.

7. *add_trafficlight*(A, A)*:* Creates a traffic light system to all roads entering the intersection of graph node A. Destroys current right-hand intersection or roundabout.

The action space presented here corresponds to the operation mode *'Junctions and roads'*. The action space corresponding to *'Lane capacities only'* is $\{1, 2\}$. For *'Lane capacities and junctions'* the space is defined as $\{1, 2, 5, 6, 7\}$, for *'Lane capacities and roads'* the set of actions is $\{1, 2, 3, 4\}$ and for *'Junctions only'* it's $\{5, 6, 7\}$. It's worth noting that in the practical implementation if the agent chooses to add a junction with two different nodes only node B will be considered. E.g. *add_roundabout*(A, B) translates to *add_roundabout*(B, B). This is a pracitcal consideration: in case of junctions the action and the endpoint matters more than the graph connection.

14 Rewarding mechanism

At each time step in the reinforcement learning agent-environment framework, the agent takes an action and as a result, the environment changes its state and returns a reward to the agent. The agent observes the environment through the state variable described in the previous section. This section will deal with the principles that will be reflected during sending the reward signal to the agent. The main directives like exact costs and traffic engineering perspectives will be further explored in the literature review. For now it's enough to deal with the main. It's worthy to note that all the reward variables are subject to manual fine-tuning so exact values are not given here.

The main goal for the agent is to build a city that is livable for humans by keeping the number of lanes and roads to the minimum at all times. If this constraint was not present, the agent could simply build 8-lane highways in streets that are a few hundred meters long. This would render being a pedestrian miserable. It's also necessary to not build roads between nodes that is not necessary from a transportation perspective. If there's a reasonable detour between two endpoints, the cars should take that road.

The reward signal has three main components, each component is based on a different principle:

1. Cost of the infrastructure: building a long, 2-way road costs more than building a small one-way street between junctions close to eachother, and this reward component reflects this. This is a one-time penalty for the reward.

2. Human factors: each reward incorporates a factor that reflects how livable a city is. If there are no traffic lights pedestrians can't cross the road. If roads are wide, long and tangled into one another the city will become hostile towards the residents. The reward component will be higher if the city is more livable, and lower if the city is less livable.

3. Vehicular factors: it's a bad design principle to have too little capacity for the number of cars that are present in the city. The better the commuting experience for the vehicles, the higher the reward. This component awards faster travel times and dynamic flow and penalizes slow travel and traffic jams.

14.1 Cost of infrastructure

Inter-node infrastructure

The cost of building a lane has to be taken into consideration. It is more expensive to add new lanes to existing infrastructure than to build the first lanes at the beginning. This is because the sidewalk has to be destructed and then rebuilt. Also traffic flow is slower in the time of the construction because of closed roads, leading to congestions. Out of the 7 possible actions of the agent 4 are in relation to building roads and lanes. The relationship between them is the following:

1. The cost of building and destroying is linearly dependent on the length of the segment. The length cost of a unit of lane is l_{lane}.

2. The cost of building a unit (e.g. meter) of one-way lane is: c_{lane}.

3. The cost of building a lane is taken as $b_{lane} = c_{lane} * l_{lane}$.

4. The cost of building a road is equivalent to building two lanes: $b_{road} = 2b_{lane} = 2 * c_{lane} * l_{lane}$.

5. The cost of destroying a lane is less than building a lane. The coefficient parameter is denoted δ: $0 < \delta < 1$ Then the cost of destroying is: $d_{lane} = \delta b_{lane} = \delta * c_{lane} * l_{lane}$.

6. Then the cost of destroying a road: $d_{road} = 2\delta b_{lane}$.

The necessary inputs are: c_{lane}, δ.

Intra-node infrastructure

The next section will describe the cost of building different types of junctions. The types of junctions in increasing order of cost are:

1. Right-hand intersection: it is the cheapest, however it is also the most unsafe. Cheap to build and to maintain. Also inexpensive to convert to and from all other types of junctions.

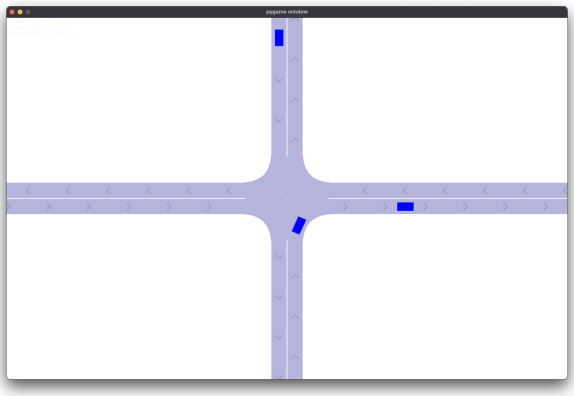

Figure 4: Right-hand intersection in the simulation window

2. Roundabout: a reasonable trade off between cost and safety, however it is difficult to destroy an intersection and build a roundabout in place of it as it requires widening the intersection. This is the best intersection in terms of dynamic traffic flow, but the worst in terms of human livability as it does not provide any safety features for pedestrians.

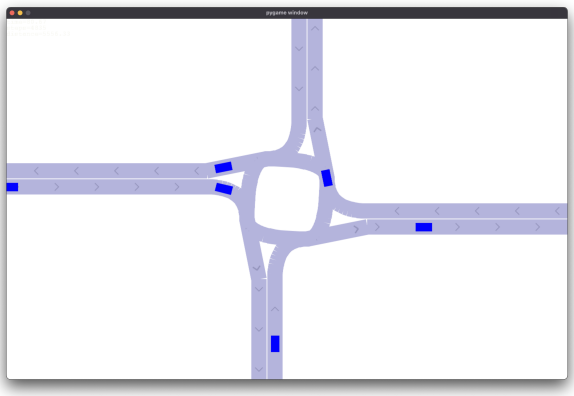

Figure 5: Roundabout in the simulation window

3. Traffic light: the most expensive and the safest type of junction is the traffic light one as it requires dedicated infrastructure and the maintenance costs are higher than in case of all the other junctions. The traffic light is the best choice

regarding human livability as it introduces an extra safety factor for humans. However traffic lights are known to cause traffic jams if the roads are not well planned.

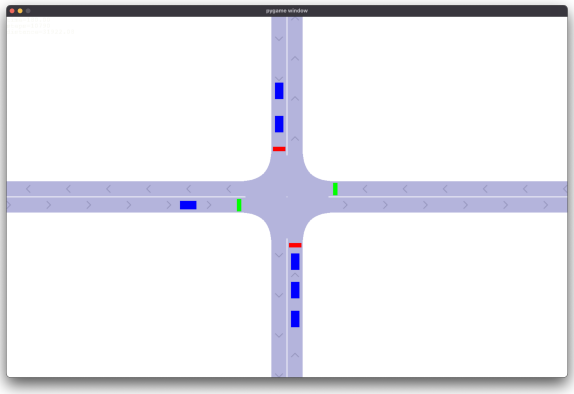

Figure 6: Traffic light intersection in the simulation window

As a graph node must be at all times assigned to a type of junction, and each type can be converted into any other type, the relationship between them can be defined by a triangle. The cost of converting from one to another can be parametrized and can be subject to change depending on how expensive building infrastructure is on a given terrain or economical environment. The conversion table is shown below:

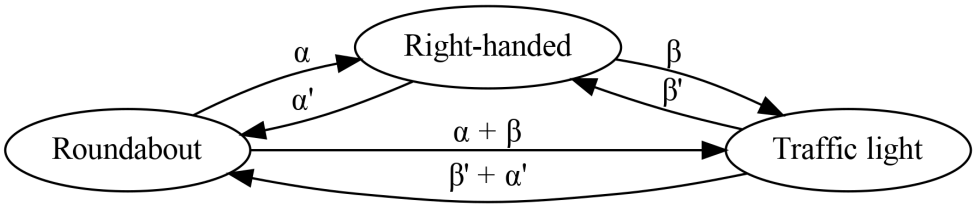

Figure 7: Conversion rates between junctions

The conversion will require the inputs to and from each type of intersection, as these are not necessarily functions on each other. Some dependency can be added later but that is independent of the current high-level model. The required inputs are: $\alpha, \alpha\prime, \beta, \beta\prime$. Each variable represents a conversion cost between two types of infrastructure. The prime version of the variable is the reverse of the same building procedure. The prime ($\prime$) will always reflect a *'destruction'* e.g. demolishing an expensive piece of infrastructure and building a cheaper one in place. The non-prime variable denotes the building of a more expensive piece of infrastructure from a cheaper one.

By default conversions are defined between roundabouts and right-handed intersections, traffic lights and right-handed intersections. Converting a traffic light into a roundabout involves first removing the traffic lights, essentially converting the junction into a right-handed intersection and then rebuilding the right-handed intersection into a roundabout. The same is true for converting a roundabout into a traffic light intersection: first the roundabout must be converted into a right-handed intersection and then traffic lights must be added to the intersection thereby finishing the conversion into a traffic light intersection. Therefore conversions between traffic lights and roundabouts are defined as $\alpha + \beta$ and $\alpha\prime + \beta\prime$.

There are limiting factors to different types of junctions e.g. how many roads it can intersect. The roundabout is famous for being able to intersect five or more roads without any problem because of the structure of it. Traffic lights can also have more roads than average coming in and out of it. For the most part right-hand intersections can handle at most four bidirectional roads. This is because human attention cannot be focused to so many places at the same time. More roads will translate to a more dangerous environment and will require highly dedicated infrastructure. It is possible, but it's rarely seen in the real world so it's not going to be considered in this research as it reflects bad design principles.

14.2 Human factors

Each training iteration the human factors get assessed according to the city map. Human factors considered in the project are:

1. How long roads are: roads too long get penalized. If we build a road to the furthest part of the city from a given junction the face of the city will be too vastly modified. Each road longer than half of the spread of the city will result in a penalty.

2. How many lanes a road is: each additional lane is more expensive to build than the previous one. This is called expansion in construction terms. If the agent builds an additional lane above a given threshold a larger cost value will be accounter for.

3. Types of junctions: roundabouts are generally penalized and traffic lights are rewarded. Roundabouts are good for vehicles as they provide dynamic intersections but they don't provide possibilities for pedestrians to cross. The traffic light intersection is the exact opposite: it allows for pedestrians to cross while blocking high-speed motor traffic. Right-hand intersections get no penalties.

14.3 Motor vehicle aspect

Each training iteration the simulation is ran for a given amount of steps.

1. Car distance and vehicles spawned: through the simulation it's measured how much distance each car has taken and how long did they go. This value is then added to the reward, so a higher value will result in a larger reward.

2. No nodes alone: if the graph is continuous the agent receives a reward to learn to not leave out any junction from the traffic planning.

3. Nodes alone: if there's a node that has no connection to any other node, a penalty is added to the reward.

14.4 Other considerations

The agent can take a number of actions and some are incorrent in a given context. For this reason a negative reward is given to the agent in the following cases:

1. The agent tries to remove a lane or road where there is currently nothing to remove (impossible).

2. The agent tries to add a lane or road where it has already reached maximum capacity (defined by n_lanes).

3. The agent tries to add a lane or road with the starting and ending nodes being the same (impossible).

4. The agent tries to add a junction where it is already built e.g. can't add a roundabout where it's already a roundabout (impossible).

The agent has to learn that this is bad behavior and define a behavior for it.

15 Network architecture

The input for the neural network is the graph of the city represented by an adjacency matrix. On each prediction pass the network has to predict three values: *start*, *end* and *action* (in this order). There were several implemented and tested architectures to achieve reward maximization such as dense networks, ensemble networks, embedding and graph-convolutional networks. Below is the architectural description of the architecture that measured the best in the testing environment.

The architecture has to predict ending node B and action a at each iteration. The starting node is first given by the environment at random, then each iteration the revious ending node will be assigned as starting node. The naive implementation

would be to use three separate neural networks predict each of the variables. This method was experimented on during training and the separate network architecture has performed poorly. This architecture is errenous in practice as two complete neural networks would have to be trained separately. The approach is redundant and can lead to information being lost and increased computation times.

In practice it's beneficial to make networks aware of eachother's predictions. This approach is reasonable as there are dependencies between the variables: $A \twoheadrightarrow B$; $A \twoheadrightarrow a$; $B \twoheadrightarrow a$; $x(A, B) \twoheadrightarrow a$. The ending node depends on the starting node, the action depends on the starting starting node and ending node and the current state of the infrastructure that is currently built between them. If these dependencies are respected the true task can be learned better by the neural network.

The activation function used during the training was ReLu with an optimizer of Adadelta and learning rate of 0.001.

End prediction

Below is the graph that is representative of the implemented deep learning model responsible for predicting the ending node based on structural information of the state and the starting node that's given in the current learning iteration. Each square node represents a deep learning layer with trainable weights and each oval node represents some transformation in the neural network.

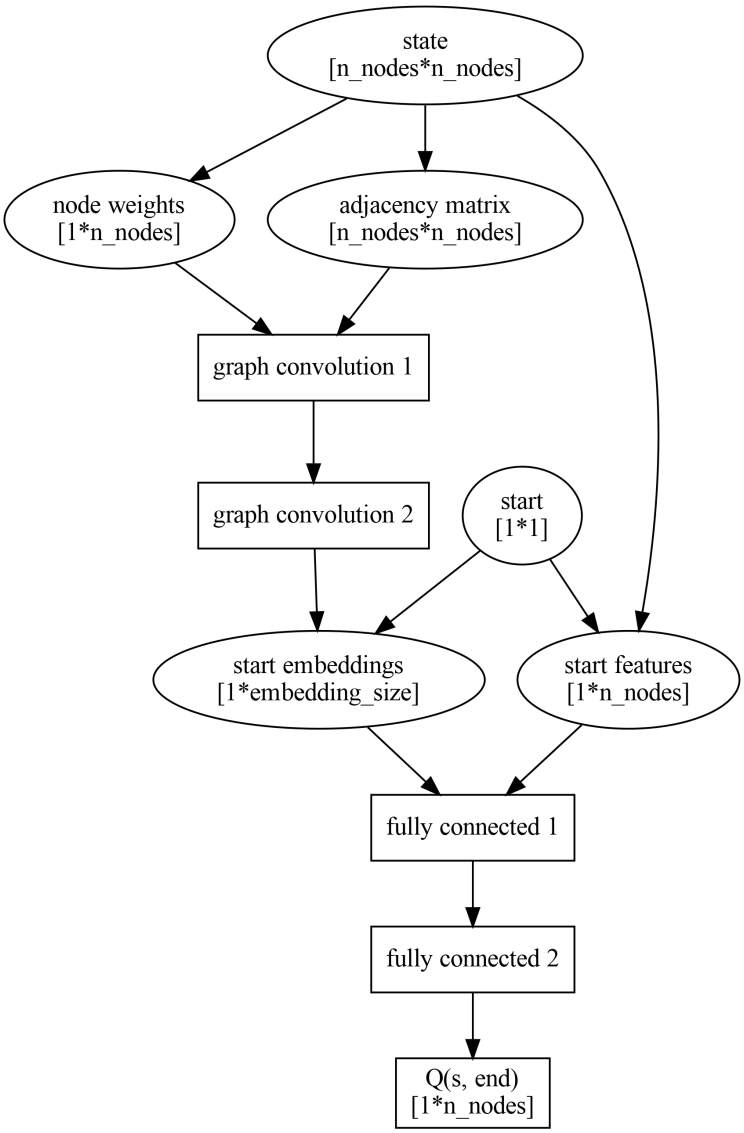

Figure 8: Graph-convolutional network to predict the ending node

The input for the network is the adjacency matrix and the starting node. First the adjacency matrix gets separated into node weights, which is the state-definition matrix's main diagonal. The adjacency matrix is the state-definition matrix with zeros on the main diagonal. The graph convolutional layers take these variables as inputs and produces an embedding for each node with a predefined size. The starting node's embedding and features from the state-definition matrix is selected

using the starting node index. The embedding vector and the feature vector are concatenated and flattened and passed to densely connected layers. The output layer produces the state-action value function which represents how good is it to select an ending node from a given starting node. The agent will set the Q-values of the ending nodes that have already been visited to $-\infty$.

Action prediction

The second neural network is used to predict the action based on the state, start and end. For this reason it's slightly more complex but the general structure is the same as before. The two networks have separably trainable weights and the same number of layers. The number of neurons in the graph convolutional and fully connected layers can be customized as preferred. During most of the experiments each of the layers in the action prediction architecture had twice as many neurons as the corresponding layer in the end prediction architecture. The structure of the GCNN is as follows:

Figure 9: Graph-convolutional network to predict the action

The inputs for the model are the state-definition matrix and the index of the starting and ending nodes. Just as before the state matrix gets separated into node weights and adjacency matrix in order to bring them to a shape that's appropriate for the graph convolutional layers. After the embeddings of the nodes have been extracted using the convolutional operations the correct embedding features are selected using the indices for start and end. The features of the start and end are selected from the matrix using the same indices. The four extracted vectors get concatenated into a single one that has the shape of $2*embedding\_size+2*n\_nodes$ and the resulting 1-dimensional vector gets passed to the fully connected layers. The amount of layers at each neural pass can be customized but during training two hidden layers were used. The output layer is used to predict the state-action value function for the action which represents how lucrative is it for the agent to be in a certain state and given two nodes for the start and end take each action. The

Q-values of the actions that cannot be taken for each start, end pair get suppressed to $-\infty$. This involves finding what the current junction is and how many lanes are currently built between the starting and ending node. If for example there are no lanes between the two junctions the agent cannot take the remove_lane action.

Part V

Results

The following section will deal with introducing the findings of the research. The machine learning task proved itself to be a very difficult one even for modern machine learning methods. Several of the most common approaches were implemented and tried in the environment however only a few of them proved successful. There were several factors that turned out to vastly influence the performance of the neural network such as costs, environmental restrictions and hard-coded rules. The deep learning environment has more than 35 parameters, not counting the ones needed to define and train the neural network. The main metric that was used to measure the performance of the models was the cumulated reward during each learning epoch. The comparison of networks in this section will include networks trained with slightly different hyperparameters, to each their own well-performing set of values.

References

[1] Mohamed A Abdel-Aty and A Essam Radwan. Modeling traffic accident occurrence and involvement. *Accident Analysis & Prevention*, 32(5):633–642, 2000.

[2] National Highway Traffic Safety Administration. National center for statistics and analysis. (2020, december). early estimate of motor vehicle traffic fatalities for the first 9 months (jan-sep) of 2020 (crash stats brief statistical summary. report no. dot hs 813 053). 2020.

[3] K Andersen and T Forr. The state of construction technology. 2018.

[4] Douglas Bates, Martin Maechler, Ben Bolker, and Steven Walker. Mixed-effects models using lme4. *J Stat Softw*, 67(1):1–48, 2015.

[5] John DeCicco, Freda Fung, and Feng An. *Global warming on the road: the climate impact of America's automobiles.* Environmental Defense, 2006.

[6] Conor K Gately, Lucy R Hutyra, and Ian Sue Wing. Cities, traffic, and co2: A multidecadal assessment of trends, drivers, and scaling relationships. *Proceedings of the National Academy of Sciences*, 112(16):4999–5004, 2015.

[7] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[8] Neil Hoult, Peter J Bennett, Ivan Stoianov, Paul Fidler, Maksimovi edo, Campbell Middleton, Nigel Graham, and Kenichi Soga. Wireless sensor networks: creating smart infrastructure. In *Proceedings of the Institution of Civil Engineers-Civil Engineering*, volume 162, pages 136–143. Thomas Telford Ltd, 2009.

[9] Susan Jette. Book review: Happy city: Transforming our lives through urban design, by charles montgomery, 2013.

[10] Roozbeh Kangari and Tetsuji Yoshida. Automation in construction. *Robotics and autonomous systems*, 6(4):327–335, 1990.

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

[13] Charles L Marohn. *Thoughts on building strong towns.* CreateSpace Independent Publishing Platform, 2012.

[14] Charles L Marohn Jr. *Strong towns: A bottom-up revolution to rebuild American prosperity*. John Wiley & Sons, 2019.

[15] Charles L Marohn Jr. *Confessions of a Recovering Engineer: Transportation for a Strong Town*. John Wiley & Sons, 2021.

[16] C Marshall, KT Boyd, and CG Moran. Injuries related to car crime: the joy-riding epidemic. *Injury*, 27(2):79–80, 1996.

[17] John Milton and Fred Mannering. The relationship among highway geometries. 1998.

[18] Peter D Norton. Street rivals: Jaywalking and the invention of the motor age street. *Technology and culture*, 48(2):331–359, 2007.

[19] Peter D Norton. *Fighting traffic: the dawn of the motor age in the American city*. Mit Press, 2011.

[20] Daniel G Parolek. *Missing Middle Housing: Thinking Big and Building Small to Respond to Today's Housing Crisis*. Island Press, 2020.

[21] John Pucher and Christian Lefevre. The united states: the car-dependent society. In *The urban transport crisis in Europe and North America*, pages 175–200. Springer, 1996.

[22] Rafael Sacks, Mark Girolami, and Ioannis Brilakis. Building information modelling, artificial intelligence and construction tech. *Developments in the Built Environment*, 4:100011, 2020.

[23] Henrikki Salo-Pontinen. Ai ethics-critical reflections on embedding ethical frameworks in ai technology. In *International Conference on Human-Computer Interaction*, pages 311–329. Springer, 2021.

[24] Aayush Sukhadia, Khush Upadhyay, Meghashree Gundeti, Smit Shah, and Manan Shah. Optimization of smart traffic governance system using artificial intelligence. *Augmented Human Research*, 5(1):1–14, 2020.

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Leaning*. 2018.

[26] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[27] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[28] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18, 2021.

List of Figures

List of Tables