Deep Reinforcement Learning for Autonomous Traffic Light Control

Deepeka Garg, Maria Chli, George Vogiatzis
School of Engineering and Applied Science
Aston University
Birmingham, UK.
e-mail: gargd@aston.ac.uk, m.chli@aston.ac.uk, g.vogiatzis@aston.ac.uk

***Abstract*—In urban areas, the efficiency of traffic flows largely depends on signal operation and expansion of the existing signal infrastructure is not feasible due to spatial, economic and environmental constraints. In this paper, we address the problem of congestion around the road intersections. We developed our traffic simulator to optimally simulate various traffic scenarios, closely related to real-world traffic situations. We contend that adaptive real-time traffic optimization is the key to improving existing infrastructure's effectiveness by enabling the traffic control system to learn, adapt and evolve according to the environment it is exposed to. We put forward a vision-based, deep reinforcement learning approach based on a policy gradient algorithm to configure traffic light control policies. The algorithm is fed real-time traffic information and aims to optimize the flows of vehicles travelling through road intersections. Our preliminary test results demonstrate that, as compared to the traffic light control methodologies based on previously proposed models, configuration of traffic light policies through this novel method is extremely beneficial.**

Keywords-component; Autonomous Traffic Control; Machine Learning; Deep Reinforcement Learning; 3d Virtual Reality Simulator.

I. INTRODUCTION

Traffic management is a major problem with significant economic and environmental repercussions. Urbanization and motorization have caused an imbalance between demand and supply of transportation and traffic infrastructure, leading to problems such as travel delays, increase of road accidents, environmental degradation and so on.

A road intersection is a shared physical space; access to this common resource must be granted intelligently to optimize the traffic throughput while ensuring safe passage of vehicles. Ever since their advent at the end of 19th century, traffic lights have been effectively used as the prime mode to grant vehicles access to the intersections, however their benefits tail off when they fail to adapt to changes in traffic flows [1].

For efficient utilization of already existing traffic-based resources, it is critically important to carry out optimization in an automated and adaptive manner, embodying characteristics such as self-configuring, self-optimizing, self-protecting and self-healing.

Leveraging to the recent advancements in the field of deep reinforcement learning, we show that following such an approach can increase the efficiency of signal-controlled traffic resources (intersections, smart lanes etc.). We propose an end-to-end traffic light control system which makes use of the raw pixels to detect the inconsistent traffic flow effectively, to determine the best set of traffic light policies to increase the traffic throughput and safety across the junction. Our system learns (without being explicitly taught), the significant visual features to accurately monitor and effectively control the traffic light signals in real-time, to optimize their performance across a range of metrics (traffic throughput, travel time, delay). Our results indicate the transition from low to high traffic throughput, as our system gradually learns to predict the best set of traffic light policies based on the current traffic conditions.

The remainder of this paper is structured as follows; Section II includes the related work in this domain. Section III covers the basic ideas of Reinforcement Learning, Deep Learning and their combination to underpin our proposed algorithm. Section IV introduces our traffic simulator. Section V specifies our model architecture specifications. Section VI covers details of our experiment and section VII discusses results and their analysis. Finally, section VIII summarizes our conclusions and discusses future work.

II. RELATED WORK

Several research works attempt to optimize traffic light performance. In [2], Dresner et al. proposed a centralized road intersection control system by making use of vehicle-to-infrastructure (V2I) communication for providing each vehicle a slot in space-time to pass through the intersection. In [3], Tachet et al. developed a comprehensive analytical framework for comparing slot-based system intersection control with traditional traffic lights. Theoretically, their results showed that transitioning from the traffic light system to a slot-based system can double the intersection throughput while significantly reducing the delay. However, their research approach requires the bulk of communication between the central controller (intersection manager) and the vehicles, which can potentially lead to significant amount of overhead in the communication network and delays due to loss or corruption of messages. Moreover, due to its centralized nature, this system is prone to single-point-of-failure bottlenecks; if the intersection manager fails, passing through the intersection would be chaotic and risky, especially in a busier junction.

Arguing about slot-based systems' susceptibility to single point of failure shortcomings, in [4] Azimi et al. proposed a distributed intersection management system, by

978-1-5386-7831-2/18/$31.00 ©2018 IEEE

creating vehicle-to-vehicle communication (V2V) for managing the movement of vehicles through the intersections. Their results showed that V2V communication can significantly reduce delays introduced by traffic lights and stop signs. However, their approach did not consider information/data packets loss within the wireless communication network, which is one of the most critical issues of wireless communication. Dropped messages between the vehicles can lead to fatal crashes as the vehicles would not be able to sense the other vehicles around them. Also, they ignore vehicle position inaccuracies, which is one of the main considerations of their protocol as every vehicle within the network depends on its position and the position of the other vehicles to gain access to the intersection.

In [5] and [6], Genders et al. proposed an adaptive traffic signal control system based on deep RL. They represented the current state of their traffic environment using a matrix indicating the presence and absence of vehicles at a certain position, which we believe is not the best state representation and does not completely exploit the potential of deep learning. In [7], Mousavi et al. followed a similar approach as ours to predict the best possible traffic signal policies. As their traffic state representation, they used raw image pixels. All these deep RL-based approaches were experimented on a commonly used traffic simulator, SUMO which is not able to optimally reflect a real-world traffic scenario. SUMO does not allow variability, most of the SUMO models only concentrate on describing a specific traffic behavior such as spontaneous traffic jams, making it inappropriate to be used for simulating complex scenes consisting of a variety of traffic situations such as peak and quiet traffic hours. Random collisions in SUMO are handled by using teleporting, which seems highly unrealistic when it comes to penalizing collisions in order to avoid the situations in the future which caused collisions in the past. Our traffic simulator is free from these limitations and is capable of effectively simulating a variety of realistic 3D traffic scenes.

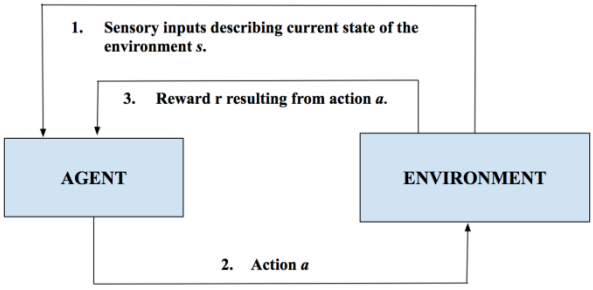

Figure 1.Basic Reinforcement Learning Mechanism Example.

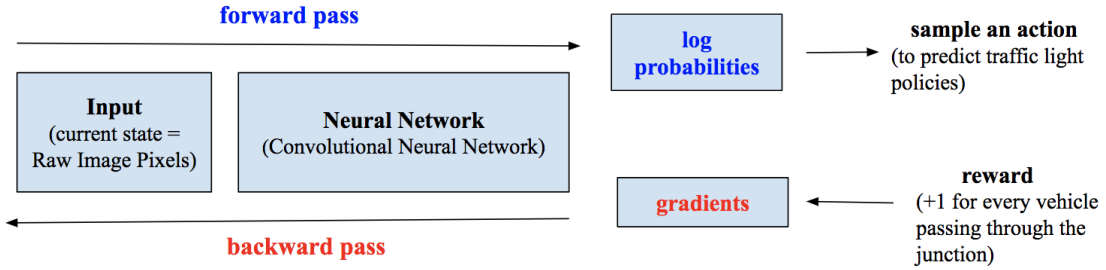

Figure 2. Policy Gradient Reinforcement Learning Pipeline

III. BACKGROUND

A. *Reinforcement Learning (RL)*

RL enables an agent to learn how to achieve a certain goal by dynamically interacting with its environment. An agent learns how to map situations to actions in order to maximize a numerical reward signal. The agent has a repertoire of possible actions and it discovers those that yield the greatest
reward at each setting by trying them out. RL is particularly useful in situations where data arrives in a continuous manner and the agent needs to adapt its behavior in real time. Figure 1 outlines the basic reinforcement learning mechanism, depicting the interactions between the agent and the environment.

A standard RL framework can be mathematically modelled as a Markov Decision Process (MDP). An MDP is defined by a five-tuple: $<S, A, T, R, \gamma>$, where S is a set of environment states, A is a set of agent actions, T is the transition function, which defines the probability of moving between the different environment states, R is the reward function and γ, $(0 \leq \gamma \leq 1)$ is known as the discount factor, which models the relevance of immediate and future rewards. The goal of the RL agent is to learn a policy $\pi: S \rightarrow A$ that maximizes the expected cumulative discounted reward. The discounted expected reward, R_t, at time t is illustrated in equation (1), where E denotes the expectation of the discounted reward and k denotes the number of actions.

$$R_t = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \quad (1)$$

B. *Deep Learning (DL)*

Deep learning is the state-of-the-art paradigm that has revolutionized representation learning, allowing a computational model to be fed with raw data in the form of

images, texts etc. and automatically discovering the data representations needed for various tasks, such as visual object recognition. A computational model composed of multiple processing layers can efficiently extract discriminative information from high-dimensional data with multiple layers of abstraction. It works through using a back-propagation algorithm to specify how a computational model should alter its internal parameters which are used to obtain a representation in each layer from the representation in the previous layer. Neural Networks are used as the above-mentioned computational models. They are initialized by a set of parameters θ, and map an input vector to an output vector through a series of hidden layers. Connections between the neural network units (neurons) of consecutive layers are known as weights (model parameters). A neural network composed of more than one hidden layer is known as a Deep Neural Network (DNN). In this research work, we implement a Convolutional Neural Network (CNN). CNNs have proven to be exceptionally powerful at visual object recognition and classification tasks. CNNs derive their name from the 'convolution operator', which captures the 2D nature of images preserving the spatial relationship between the pixels.

C. *Deep Reinforcement Learning (DRL)*

Deep Reinforcement Learning combines RL with Deep Learning. Previously-made attempts to combine RL with neural networks failed due to unstable learning in high-dimensional state and action spaces. To address these instability issues, Deep Learning in conjunction with RL was introduced. Deep Neural Networks (DNNs) are used as function approximators in a deep RL framework. Deep learning enables an RL agent to scale to decision-making problems which were previously considered intractable.

Development of algorithms that were able to learn to play a range of Atari 2600 video games directly from the raw image pixels at a superhuman level, revolutionized the field of DRL [8]. DRL worked extremely well on a variety of video games, which motivated us to apply these algorithms in creating an autonomous traffic light system that is capable of learning how to adapt to the different traffic conditions, such as rush hours, different times of the day and sudden unexpected changes to traffic conditions, giving priority to emergency vehicles etc.

D. *Policy Gradient (PG) Reinforcement Learning*

The standard RL approach is approximating a value function and discovering a policy from it, known as Q-learning. But in non-deterministic environments, this approach has proven to be theoretically intractable. Policy gradient is preferred when dealing with stochastic environments, because it is an end-to-end approach including an explicit policy which directly optimizes the expected reward.

The policy is defined as the mapping from the state to the action probabilities, where $\pi(s, a)$ is the probability of taking an action a in state s under the policy π. The policy is a function represented by a neural network, whose input is some representation of the state, outputs are action selection probabilities and weights are the policy parameters. Let θ represent the vector of policy parameters and ρ represent the performance of the corresponding policy (discounted expected reward per step). Then, as per the policy gradient approach, the policy parameters are updated approximately proportionally to the gradient, as per equation (2).

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta}, \tag{2}$$

where α is a step size. Usually, θ can be assured to converge to a locally optimal policy in the performance measure ρ. Basically, our goal is to maximise the reward under the probability distribution $\pi(a_t|s_t;\theta)$.

Figure 2 illustrates the PG mechanism pipeline. At every time-step, some representation of the current state of the environment is fed into to the neural network as input. In our work, state representation is in the form of images. In the forward pass, the neural network calculates the probabilities of the predefined actions, an action is sampled from the action probability distribution. Based on the received rewards, gradients are computed in the backward pass as per equation (3), $J(\theta)$ denotes the loss function.

$$\nabla_\theta J(\theta) \approx \sum_i (\sum_t \nabla_\theta log \pi_\theta(\mathrm{a}_t^i|\mathrm{s}_t^i))(\sum_t r(\mathrm{s}_t^i, \mathrm{a}_t^i)) \tag{3}$$

Lastly, the policy is updated in the direction of the gradient, as shown in equation (4) to encourage the actions leading to good outcomes and discourage the ones leading to the bad outcomes.

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{4}$$

This section gave a brief introduction to the key elements that constitute our research approach. The following section introduces our simulation environment.

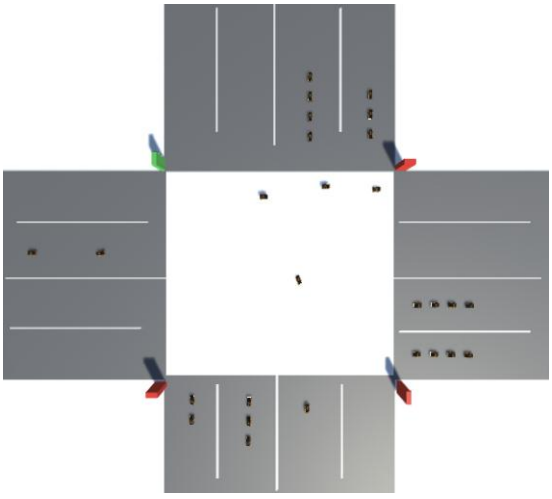

Figure 3: Our Simulator

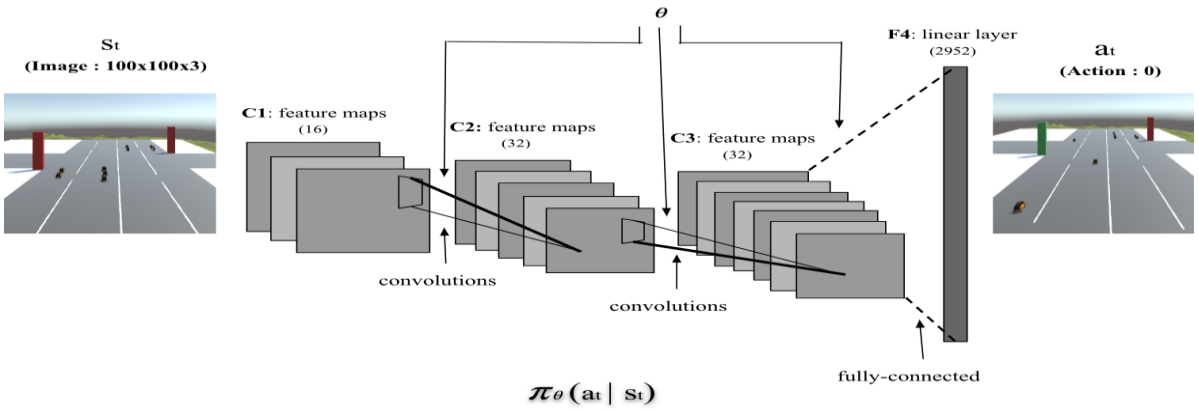

Figure 4. Our Architecture

IV. Our Simulator and it's Setup

In the transportation and traffic area of research, simulations are typically used as the first step in the protocol development. To simulate traffic scenarios, the transportation research community uses a variety of traffic simulators like SUMO [9], VISSIM [10], CORSIM [11] etc. Most of the existing traffic simulators are not realistic to emulate real world traffic. These simulation tools often suffer from various shortcomings, such as no collision count, having the speed of vehicles constant across the intersections, no dynamic generation of vehicles within the simulation environment etc. We believe that these shortcomings are significant and cannot be ignored, especially if the potential application of developed traffic protocols is to be deployed in a real-life setting.

To realistically validate our research idea, we built a traffic simulator on a 3d virtual reality software, Unity3d to create a simulation environment closely based on the real-world traffic specifications, such as vehicle arrival rate conforms to the random distribution to mimic the real-world traffic scenarios. For an efficient vehicle trajectory control, the parameters; such as maximum speed, maximum acceleration and deceleration rate are most important, our simulator has these parameters configured based on real world vehicle specifications. Our simulator is able to seamlessly switch between a variety of different traffic environments, such as rush/quiet hours etc., which is not feasible while using conventional traffic simulators. Figure 3 shows a screenshot of our simulator's graphical display.

Most importantly, our simulator enables us to capture still pictures and video footage, in much the same way as is possible in a real-world traffic scenario, to be used to analyze and determine traffic policies.

We created a real-time interface between Unity3d and Python using socket programming for bilateral data exchange between them. Our policy network, outlined in figure 2, is implemented in python. The section on architecture, below, describes the data exchange mechanism between the two softwares in detail.

V. Our Architecture

Our architecture is composed of two components; Unity3d and Python. As stated in the previous section, we get our road intersection environment's graphical display from unity3d, while our policy network is implemented on python. These two components of our architecture interact with each other using socket programming, creating a fixed and a logical connection between server and client. In our work, python acts as a server and unity3d acts as a client. At every time-step, unity3d transmits the representation of the current state (s) of the environment to the policy network (implemented on pytorch, a python platform) via socket interface and in return, receives an action (a) to decide the configuration of traffic light policies and sends back a numerical reward (r), based on the quality of action taken.

Figure 4 illustrates our architecture. Our policy network which implements our traffic light controller agent, is a 4-layer DNN composed of 3 convolutional layers (C1 with 16 output channels, C2 with 32 output channels and C3 with 32 output channels) and 1 fully-connected layer (F4 with 2952 neurons). The current state (s_t) of traffic light junction, is fed into the policy network. θ denotes our policy network parameters, which are randomly initialized in the beginning. From the action probabilities distribution computed from the underlying policy π, an action (a_t) is sampled and implemented. Our action space includes two discrete actions {0, 1}, indicating what lane go give green to. Every action produces a reward, based on which, the gradient is computed and policy parameters (θ) are updated to continuously improve the policy (π), until an optimal policy is found. Algorithm 1 outlines our policy gradient algorithm.

Algorithm 1:

1. Initialize model parameters, θ with random values
2. Initialize time-step counter t = 0
3. For each time-step starting from t = t_{start}, repeat:
   - 3.1 initialize s with current visual of the intersection
   - 3.2 select an action a according to policy, $\pi_\theta(a_t|s_t)$
   - 3.3 observe reward r and next state s'
   - 3.4 compute gradients, according to equation (3),

3.5 update gradients, according to equation (4).

4. Until t - t_{start} == M (max time-step)

5. **End.**

VI. Experiment

As a first set of experiments, shown in figure 4, we implemented an intersection of two-lane traffic scenario with conflicting vehicular traffic such that vehicles from any one lane are granted access to the intersection at a time, while vehicles from the other lane have to wait. We applied two signal phases (red and green) with no turning movements. To optimize the traffic throughput and make our policy network learn appropriately, every vehicle passing through the junction is given a reward of +1.

We update policy network parameters (θ), after every 10 simulations (batch_size). One episode of our simulation is made of 100 time-steps. We compute the gradients for every episode and keep accumulating them and update the network parameters after batch of 10 simulations.

To make the policy network learn and evolve continuously and efficiently, it is required to face a variety of different state situations. To make things more challenging for our policy network, we kept the vehicle density same for both the lanes thus creating less variation in our environment. We compare our policy gradient-based traffic light approach with two baselines; traditional fixed-traffic lights and uncontrolled junction (with no traffic lights, first-come-first-served basis).

VII. Results and Discussion

Our test results revealed that after running our simulator for about 600,000 vehicles, our policy gradient algorithm converged. Even with minimum variability in the simulation environment, we obtained an encouraging set of results. Our policy network performed as well as the fixed-traffic lights. Since as a preliminary test, we implemented our approach on a simple, uncomplicated traffic junction, we expected our policy network to reach the performance of the fixed-traffic lights, as shown in figure 5. Our traffic light controller agent could successfully sense the current state of the intersection and learn to take appropriate actions to increase the traffic throughput through the intersection. This is a positive result and we expect our system to crucially outperform fixed-traffic lights in situations when there is a sudden change in traffic conditions, flows and densities, or an accident causing the bottleneck.

VIII. Conclusion and Future Work

We introduced an autonomous traffic light control system, based on deep RL. We experimented our research idea on a novel traffic simulator to realistically frame our research problem. We showed our approach performed well for a simple traffic light intersection scenario.

To verify the robustness of our research approach and its applicability to dynamically varying and diverse traffic conditions, we plan to extend our approach to more complicated road intersections with multiple lanes, like the one shown in figure 3. To collectively optimize traffic through multiple road intersections, we will create real time coordination between multiple road intersections for information sharing and traffic light operation negotiation. We will ensure that our system can withstand various traffic-related stresses like degradation of camera output, single point of failure of policy gradient-based traffic light control system and unreliable communication between communicating junctions in a multiple-junction traffic light scenario.

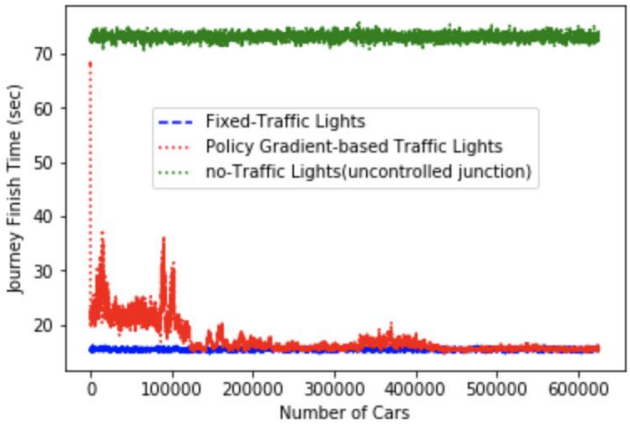

Figure 5. Learning curve showing vehicles' journey finish time for PG-based approach versus fixed-traffic lights and no-traffic lights.

References

1. C. Priemer, B. Friedrich, "A decentralized adaptive traffic signal control using V2I communication data, Intelligent Transportation Systems," ITSC '09. 12th International IEEE Conference on, vol., no., pp. 1-6, 4-7 Oct. 2009.
2. K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," J. Artif. Intell. Res., vol. 31, no. 1, pp. 591–656, Jan. 2008.
3. R. Tachet P Santi, S Sobolevsky, L. Reyes-Castro, E. Frazzoli, D. Helbing and C. Ratti, "Revisiting street intersections using slot-based systems," PLOS ONE, vol. 11, no. 3, p. e0149607, 2016.
4. R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks.," in SAE World Congress, 2012.
5. E. van der Pol, F.A. Oliehoek, 'Coordinated deep reinforcement learners for traffic light control', master's thesis, University of Amsterdam, August 2016.
6. W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," CoRR abs/1611.01142, 2016.
7. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policygradient and value-function-based reinforcement learning," IET Intelligent Transport Systems, vol. 11, pp.417-423, 2017.
8. V. Mnih, K. Koray, D. Silver, et al., "Human-level control through deep reinforcement learning", Nature, vol. 518, no. 7540, pp. 529-533, 2015.
9. M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMOsimulation of urban mobility—An overview," in Proc. 3rd Int. Conf. SIMUL, Barcelona, Spain, Oct. 2011, pp. 55–60.
10. L. Bloomberg, J. Dale, "A comparison of the VISSIM and CORSIM traffic simulation models", Institute of Transportation Engineers, 2000.
11. S. Boxill, and L. Yu, "An evaluation of traffic simulation models for supporting ITS development," (Technical Report 167602-1). Texas Southern University, October 2000.