Using a Deep Reinforcement Learning Agent for Traffic Signal Control

Wade Genders[a], Saiedeh Razavi[b]

***Abstract*—Ensuring transportation systems are efficient is a priority for modern society. Technological advances have made it possible for transportation systems to collect large volumes of varied data on an unprecedented scale. We propose a traffic signal control system which takes advantage of this new, high quality data, with minimal abstraction compared to other proposed systems. We apply modern deep reinforcement learning methods to build a truly adaptive traffic signal control agent in the traffic microsimulator SUMO. We propose a new state space, the discrete traffic state encoding, which is information dense. The discrete traffic state encoding is used as input to a deep convolutional neural network, trained using Q-learning with experience replay. Our agent was compared against a one hidden layer neural network traffic signal control agent and reduces average cumulative delay by 82%, average queue length by 66% and average travel time by 20%.**

***Index Terms*—Traffic control, Agent-based modeling, Adaptive systems, Machine learning, Artificial Neural Networks, Simulation**

I. Introduction

MODERN society relies on its many transportation systems for the movement of individuals, goods and services. Ensuring vehicles can move efficiently from their origin to destination is desirable by all. However, increasing population, and subsequent vehicle ownership, has increased the demand of road infrastructure often beyond its capacity, resulting in congestion, travel delays and unnecessary vehicle emissions. To address this problem, two types of solutions are possible. The first is to increase capacity by expanding road infrastructure, however this can be expensive, protracted and decrease capacity in the short term. The second solution is to increase the efficiency of existing infrastructure and the systems that govern them, such as traffic signal controllers (TSC). We advocate this second solution, by utilizing recent advancements from the domain of artificial intelligence [1] to develop a new traffic signal controller.

We define the traffic signal control problem as follows; given the state of traffic at an intersection, what is the optimal traffic signal phase and sequence that should be enacted? Many systems have been proposed that utilize new sensors, particularly reinforcement learning for traffic signal control, however they do not take full advantage of the available data. We propose a deep artificial neural network as a traffic signal control agent (TSCA), trained using reinforcement learning, that strives to solve the traffic signal control problem by developing an optimal control policy.

Reinforcement learning is a machine learning paradigm where an agent seeks to maximize cumulative reward by developing a state-action policy through repeated interaction with its environment. Reinforcement learning agents achieve optimal control with respect to a defined reward by developing an optimal state-action policy. Function approximators, such as artificial neural networks, have been used in reinforcement learning to approximate value functions when the agent's representation of the environment, or state space, becomes too large [2]. Convolutional neural networks, a specific type of network architecture, are inspired by biological research on the animal visual cortex [3][4] and have displayed impressive performance [5]. They apply the mathematical convolution operation between various filters and the layer input to produce feature maps. Convolutional networks are advantageous because minimal input pre-processing is required and they can develop their own features. We develop a deep Q-network traffic signal control agent (DQTSCA), with the action-value function modeled as a deep convolutional neural network trained using reinforcement learning in a traffic microsimulator, SUMO, on an isolated intersection.

Reinforcement learning is a suitable technique for attempting to solve the traffic signal control problem, as it elegantly represents the elements of the problem - agent (traffic signal controller), environment (state of traffic) and actions (traffic signals). Previous research using reinforcement learning for traffic signal control has yielded impressive results [6][7][8], yet we perceive areas for improvement. We propose a new state space definition, the discrete traffic state encoding (DTSE), as an improved representation of traffic, as it contains more relevant information compared to previous research's state space definitions. The DTSE is proposed as it is information dense; the convolutional neural network is required to take advantage of the information dense state. The DTSE will allow the convolutional neural network to perceive more relevant traffic information than previous research, extract useful features and develop high-level state representations. The agent can then achieve optimal control by choosing the actions with the highest value, or maximum expected cumulative reward.

The succeeding sections are organized as follows: Section II details research conducted in the domain of traffic signal control and reinforcement learning, Section III describes the proposed DQTSCA and defines the state space, action space and reward, Section IV details the tools used to implement the

[a] Ph.D. Student, Department of Civil Engineering, McMaster University, Hamilton, Ontario, L8S 4L8 Canada email: genderwt@mcmaster.ca

[b] Associate Professor, Chair in Heavy Construction, Department of Civil Engineering, McMaster University, Hamilton, Ontario, L8S 4L8 Canada email: razavi@mcmaster.ca

proposed agent and describes its training, Section V discusses the results and performance of the agent and Section VI summarizes the research conducted and provides ideas for future work.

II. Literature Review

Significant research has been conducted using reinforcement learning for traffic signal control. Early efforts were limited by simple simulations and a lack of computational power [9][10][11][12]. Beginning in the early 2000's, continuous improvements in both of these areas have created a variety of simulation tools that are increasingly complex and realistic. Traffic microsimulators are the most popular tool used by traffic researchers, as they model individual vehicles as distinct entities and can reproduce real-world traffic behavior such as shockwaves. Research conducted has differed in reinforcement learning type, state space definition, action space definition, reward definition, simulator, traffic network geometry and vehicle generation model. Previous research efforts have defined the state space as some attribute of traffic, the number of queued vehicles [10][12][13][14] and traffic flow [15][6] the most popular. The action space has been defined as all available signal phases [15][8] or restricted to green phases only [6][13][14]. The most common reward definitions are change in delay [15][8] and change in queued vehicles [6][13][14]. For a comprehensive review of reinforcement learning traffic signal control research, the reader is referred to [16] and [17].

Regarding previous research, the following observations can be made. First, the majority of state definitions are abstractions of the traffic state which omit relevant information. A reinforcement learning agent must first observe the state of the environment before it can act, if useful information is missing, it is unlikely to be able to act optimally. For example, if the state space is defined as the number of queued vehicles at the intersection, this ignores all of the moving vehicles, as well as the queued vehicles' lane and queue position. We believe the state space definition should include as much relevant information about the traffic state as possible, including vehicles' location and speed, thus our proposal of the DTSE, formally defined in Section III. We recognize that in practice it may be difficult for a TSCA to observe the state of all vehicles' location and speed, but we will defend this assumption in succeeding sections. However, some previous research has proposed a similar, less abstracted, yet limited, state definition [9], from which our research acknowledges their contribution and seeks to extend beyond their efforts.

Second, the TSCA should be given as much action autonomy as possible, therefore it must be recognized that defining the action space as choosing between fixed sequences of signal phases is limiting. For example, if we define that an advance left green signal phase must always precede a through green signal phase, this assumes the optimal policy follows such a sequence. However, it is conceivable that the optimal action given a certain traffic state is to have an advance left green signal phase succeed a through green signal phase. Much of the previous research has constrained the agent's action in such a way; our action space definition seeks to endow the agent with a higher degree of autonomy in an attempt to learn the optimal policy.

Finally, all previous research have used computer simulations, as real-world experimentation is infeasible for various reasons.The majority of research assumes vehicle generation can be modeled as a Poisson process, which relies upon the negative exponential distribution, to model the time between vehicle generation events. We propose in subsequent sections that the negative exponential is not the best distribution to model real traffic, as empirical research has shown other distributions to more accurately model different vehicle generation flow rates.

III. Proposed System

Attempting to solve the traffic signal control problem using reinforcement learning requires a formulation of the problem in the language of reinforcement learning, specifically, defining a state space S, an action space A and a reward R.

A. State Space

We propose the DTSE as the appropriate state space S in this research, inspired by a common technique in computing of discretization and quantization of continuous entities. For each lane approaching the intersection, the DTSE discretizes a length l of the lane segment, beginning at the stop line, into cells of length c. The selection of c will change the behavior of system. If c is many times larger than the average vehicle length, the individual dynamics of each vehicle will be lost, however computational cost will be reduced. If c is much smaller than the average vehicle length, the individual vehicle dynamics will be retained, however the computational cost will increase, perhaps unnecessarily. We mention the selection of c is important, however for this research we select c in a simplified manner in an attempt to evaluate the proposed system.

The DTSE is composed of three vectors, the first representing the presence of a vehicle or not in the cell, the second the speed of the vehicle and the third the current traffic signal phase (i.e., the most recent action selected). The addition of second speed vector is an extension beyond [9], as their state definition only consists of a vector representing the presence of a vehicle. Therefore, the state of traffic at an intersection with n lanes is formally defined as the DTSE, where $S \in (\mathbb{B} \times \mathbb{R})^{\frac{l}{c} \times n} \times P$ and P represents the current traffic signal phase. At time t, the agent observes the traffic state (i.e., the DTSE) as $s_t \in S$. A representation of the DTSE can be seen in Fig. 1, with triangles representing vehicles traveling from left to right. In Fig. 1, Fig. 1 (a) shows simulated vehicles approaching the intersection, Fig. 1 (b) is the Boolean-valued vector of the DTSE, encoding the presence or absence of a value and Fig. 1 (c) is the real-valued vector of the DTSE, encoding the normalized speed.

The motivation behind the DTSE is to retain useful information. If the agent is to discover the optimal policy, it must discover the optimal actions for any given state; having knowledge of approaching vehicle's speed and position is conjectured to be superior to only the number of queued vehicles

(a)

(b)

0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1	0	1	1
0	1	0	0	0	0	1	0	0	1	0	1

(c)

0	0	0	0	0	0	0	0	0.2	0	0	0
0.8	0	0	0	0.5	0	0	0	0	0	0	0
0	0.6	0	0	0	0	0.4	0	0	0	0	0

Fig. 1: Example of simulated traffic (a) with corresponding Boolean- (b) and real-valued DTSE vectors (c).

or vehicle flow. The first vector's elements are Boolean-valued, with a one representing the presence of a vehicle and a zero representing the absence of a vehicle. The second vector's elements are real numbers and represent the vehicle's speed, normalized by the speed limit. Each element of P represents a different traffic phase and all elements of P are zero except for the current phase, which is one, therefore $P \in \mathbb{B}^{|A|}$.

Technologies over the last decade have made gathering information required for the DTSE possible. Video cameras [18] are becoming more common as sensor devices at intersections and vehicles with wireless communication capabilities (i.e., Connected Vehicles [19]) are expected to be deployed in the near future. Ultimately, the DTSE is sensor agnostic, the means by which the state information is gathered, be it vision, wireless or otherwise, is irrelevant to creating the DTSE. The flexibility in generating the DTSE should be seen as an advantage of the system.

B. Action Space

After the agent has observed the state of the environment, it must choose one action from the set of all available actions. In this research, the agent's possible actions are the traffic signal phase configurations (i.e., the combination of traffic lights controlling individual lanes for the entire intersection). For simplicity and human comprehension, each action is assigned a compass direction indicating the approaching lanes' traffic signal phases (i.e., the color of the traffic signal lights) and abbreviated for brevity. For explicit clarity, a green traffic signal phase means vehicles can proceed through the intersection, yellow cautions vehicles to slow down and prepare to stop and red means vehicles should stop and not proceed through the intersection. The possible actions are North-South Green (NSG), East-West Green (EWG), North-South Advance Left Green (NSLG), East-West Advance Left Green (EWLG). Note, for any given action, it is implied that the omitted compass direction traffic signals are red (e.g., East-West Green means that all North-South traffic signals are red).

Formally the set of all possible actions A is defined as $A = \{\text{NSG, EWG, NSLG, EWLG}\}$. Therefore, at time t, the agent chooses an action a_t, where $a_t \in A$. However, when an agent chooses an action, it may not be immediately enacted. To ensure safe control of the intersection, additional traffic signal phase configurations may precede the chosen action. Instead of immediately transitioning from the current traffic signal phase to the selected action, a sequence of intermediate traffic signal phases dependent on the current phase and chosen action may be necessary. All possible action transition sequences to transition from the current traffic signal to the chosen action are shown in Table I. Note the addition of the North-South Yellow (NSY) and East-West Yellow (EWY) and All Red (R) traffic signal configurations, which cannot be chosen explicitly as actions, but are part of some traffic signal transition sequences. The yellow and red phases are necessary for safety reasons, as they slow down and stop traffic so that succeding green phases may be enacted.

C. Reward

The final element of reinforcement learning, after the agent has observed the state of the environment s_t, chosen an action a_t, and performed it, is receiving the reward. The reward is one element that differentiates reinforcement learning from other types of machine learning; developing a state-action policy which maximizes cumulative long-term reward is what the agent seeks. Compared to other types of machine learning, in which correct actions are given by instruction, reinforcement learning has the agent evaluate actions by interacting with the environment. How to select the appropriate reward for a given task is an unanswered problem in traditional reinforcement learning[1]. It would be desirable if the agent could choose its own reward, instead of requiring an expert to define it, and is therefore a goal of many active researchers.

In the context of traffic signal control, various rewards have been proposed, such as change in number of queued vehicles, change in cumulative vehicle delay and change in vehicle throughput. The reward $r_{t+1} \in \mathbb{R}$ is a consequence of enacting a selected action from a specific state. In this research, we define the reward as change in cumulative vehicle delay between actions. This allows for the reward to be positive or negative, meaning the agent can be punished ($r_{t+1} < 0$ for increase in delay) or rewarded ($r_{t+1} > 0$ for decrease in delay). The use of the subscript $t+1$ is intentional, to emphasize the temporal relationship between taking action a_t in state s_t, as the reward succeeds these two events. In addition to receiving a reward from the environment, the agent has the opportunity to observe the new state of the environment s_{t+1}, which was influenced by its most recent action. With this new state observation, a new action can be chosen and subsequently a new reward received. This cycle can be continued indefinitely or stopped according to some criteria, depending on the reinforcement learning task at hand.

D. Agent

In reinforcement learning, the agent is the entity that learns by interacting with the environment. We model the agent controlling the traffic signals as a deep convolutional Q-network [1]. Artificial neural networks are mathematical functions inspired by biological neural networks (i.e., brains) that are appealing for their function approximation capabilities.

[1]See inverse reinforcement or apprenticeship learning.

TABLE I: Traffic Signal Phase Action Transitions

		Selected Action			
		NSG	**EWG**	**NSLG**	**EWLG**
Current Traffic Signal Phase	**NSG**	-	{NSY, R}	{NSY}	{NSY, R}
	EWG	{EWY, R}	-	{EWY, R}	{EWY}
	NSLG	-	{NSY, R}	-	{NSY, R}
	EWLG	{EWY}	-	{EWY. R}	-

Many problems in machine learning can suffer from the curse of dimensionality, which is when the dimensionality of the data increases, the training and computational resources required grow exponentially. Artificial neural networks have the capability to generalize from what they have learned, weakening the problems posed by the curse of dimensionality. Convolutional neural networks are a variant of artificial neural networks inspired by biological research that emulate the architecture of the animal visual cortex [3][4], making them adept at perception tasks.

Most artificial neural networks require data pre-processing, where features of the data are determined by experts. Features are measurable aspects of the data deemed important to the present machine learning task. Expert-crafted features require assumptions to be made about the data that may or may not be true. In the context of traffic signal control, examples of expert-crafted features are queue length or average vehicle flow. These features are abstractions of the individual vehicles behavior that have been extracted and deemed important by experts for solving the traffic signal control problem using reinforcement learning. However, because they are abstractions, we argue important information is lost and the potential for learning is diminished. If only queue length is used, this assumes all vehicles not in a queue are irrelevant to developing an optimal traffic signal control policy - a spurious claim. Similarly, average flow is a historical metric calculated over some time interval, yielding a very coarse approximation of the current traffic state that ignores and abstracts away useful information. Convolutional neural networks are advantageous because they develop their own features from the data. The DTSE is proposed because it is a lesser abstraction of the traffic state than queue length or average flow and the convolutional neural network can take advantage of its information rich nature.

The depth of a deep neural network refers to the fact that there is more than one hidden computational layer of neurons. Additional layers in a network allow it to develop features of features, transforming low-level features of the data to high-level ones, potentially increasing network performance. The combination of the DTSE and the convolutional neural network allow for the creation of a truly adaptive traffic signal controller.

The DQTSCA's architecture is first two identical networks receiving different inputs. Each network receives a different input vector from the DTSE - one the real-valued vector, the other the Boolean-valued vector. The first layer of each network is a convolutional layer with 16 filters of size 4x4 applied with stride 2 using rectifier nonlinear activation functions. The second layer of each network is a convolutional layer with 32 filters of size 2x2 applied using rectifier nonlinear activation functions. The outputs of these two networks and the vector P, representing the current traffic signal phase, are combined and used as input to two fully-connected layers of 128 and then 64 neurons with rectifier nonlinear activation functions. The output layer is $|A|$ (i.e., four) neurons with linear activation functions.

The reinforcement learning algorithm used in this research is Q-Learning [20], which is used to develop an optimal action-selection policy. The optimal policy is achieved by using the convolutional neural network to approximate the action-value function. The action-value function maps states to action utilities (i.e., what is the value of each action from a given state). Values represent long-term reward. If an action has a high value, enacting it means reaping future reward, although potentially not immediate reward. We define the deep convolutional neural network as the action-value function $\eta : X \mapsto Y$, where $X \in S$ and $Y \in \mathbb{R}^{|A|}$, with Y representing the action-values $\forall A$. At time t, the input x_t to the network is $x_t = s_t$. The output y_t is a vector containing all the action-values, with y_{t,a_t} denoting the action-value of a_t. After the action-value function has been sufficiently learned, the optimal policy can be determined by selecting the action with the highest value given s_t. The basis of Q-learning is the value iteration update (Q-update), defined in (1).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Big(r_{t+1} + \gamma \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)\Big) \tag{1}$$

Where the learning rate α controls the degree to which new action-value estimates are weighted against old estimates and the discount factor γ determines how immediate rewards are weighted against future rewards. Both the learning rate and the discount factor are parameters of Q-learning and are $\gamma, \alpha \in [0, 1]$. To use the Q-update to train the deep convolutional network, a variation of (1) is used, defined in (2).

$$\begin{aligned} x_t &= s_t \\ x_{t+1} &= s_{t+1} \\ y_t &= \eta(x_t) \\ y_{t,a_t} &= r_{t+1} + \gamma \max\Big(\eta(x_{t+1})\Big) \end{aligned} \tag{2}$$

After (2) has been computed, the deep convolutional network can be trained and the weights θ updated using gradient descent with x_t as the network input and y_t as the output. We use the RMSprop [21] gradient descent algorithm with an α of 0.00025 and a γ of 0.95 to train the network. Once the deep convolutional neural network has sufficiently approximated the action-value function, optimal control is achieved by selecting the action with the highest value given the current state.

A major problem in any reinforcement learning task is the action-selection policy while learning; whether to take exploratory action and potentially learn more, or to take exploitative action and attempt to reap the most reward given what has been learned so far. The explore-exploit tradeoff is an active area of research in reinforcement learning with many proposed solutions. We implement the simple, yet effective, decreasing ϵ-greedy exploration policy, which selects a random action (explore) with a probability ϵ and selects the action with the highest value (exploit) with a probability 1-ϵ. The value of ϵ decreases as training epochs progress according to (3).

$$\boxed{\epsilon_n = 1.0 - \frac{n}{N}} \tag{3}$$

Where n is the current training epoch and N is the total number of training epochs Initially, $\epsilon = 1.0$, meaning the agent exclusively explores, however, as training progresses, the agent increasingly exploits what it has learned, until it exclusively exploits.

IV. Experimental Setup and Training

All experiments were conducted using the traffic microsimulator SUMO v0.22 [22]. SUMO provides an application programming interface (API) in the Python programming language, by which custom functionality can be implemented in the traffic simulation. We used the SUMO Python API and custom code to implement the DQTSCA. The artificial neural network was implemented using Keras [23] and Theano [24] Python libraries. Additional optimized functionality was provided by NumPy and SciPy [25] libraries. For the DTSE parameters, we define l as 75 m and c as 5 m. We train for 1 600 training epochs, where each epoch is 1.25 hours of simulated traffic. The simulations were executed on a desktop computer with a 3.40 GHz i7-2600 CPU, 8GB of RAM running Ubuntu 14.04. The length of the agent's actions (i.e., NSG, EWG, NSLG, EWLG) are two seconds and the transition phases (i.e., R, NSY, EWY) are five seconds.

The intersection geometry is four lanes approaching the intersection from the compass directions (i.e., North, South, East and West) connected to four outgoing lanes from the intersection. The traffic movements for each approach are as follows: the inner lane is left turn only, the two middle lanes are through lanes and the outer lane is through and right turning. All lanes are 750 meters in length, from the vehicle origin to the intersection stop line.

The method by which vehicles are generated and released into the network greatly influences the quality of any traffic simulation. The most popular vehicle generation method is to randomly sample from a probability distribution numbers that represent vehicle headway times, or the time interval between vehicles. This research does not break from this method entirely, however we strive to implement a nuanced version which better models real-world traffic. Empirical research has shown that different vehicle flow rates are suitably approximated by different probability distributions [26][27]. Instead of using a negative exponential distribution for all flow rates and modifying its rate parameter, we use different distributions for different flow rates, shown in Table II. The Inverse Weibull distribution is used for generating left and right turning traffic and the Burr distribution is used for generating through traffic.

The agent is trained using a biologically inspired process known as experience replay [28][29][30]. Instead of training after every individual state, action, reward, state sequence, the agent stores the experience, defined $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$, in an experience memory M for periodic, randomized batch training. The training pseudocode is presented in Algorithm 1 and 2. This research takes advantage of the multithreading capabilities of modern computers, running multiple traffic simulations in parallel. Each thread is running Algorithm 2 and generating different experiences for use in the experience replay.

Algorithm 1: Deep reinforcement learning traffic signal control agent experience replay

```
Initialize neural network agent $\eta$ with random weights $\theta$ on main agent
Copy main agent weights $\theta$ to all thread agents
For epoch=1 to $N$ do
  Copy main agent weights $\theta$ to all thread agents
  In parallel run Algorithm 2 on threads
    While all threads not finished do
      If buffer == batch_size do
        Append buffer to $M$, clear buffer
        Randomly sample batch_size experiences, from $M$
        Batch train main agent using (2)
  If epoch mod(exp_refill) == 0 do
    Clear $M$
    While len($M$) < min_size do
      In parallel run Algorithm 2 on threads
        Append buffer to $M$
```

Algorithm 2: Thread Traffic Simulation

```
For $t$=1 to sim_len do
  Observe DTSE, $s_t$
  Select random action $a_t$ with probability $\epsilon$,
  else select $a_t = \max\eta(s_t)$
  Implement selected $a_t$, increment simulation,
  observe reward $r_{t+1}$ and $s_{t+1}$,
  If len($M$) == max_size do
    delete $M$[0]
  Append $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ to buffer
```

The *buffer* in Algorithm 1 and 2 temporarily stores the most recent experiences until it reaches *batch_size*, at which point it is appended to *M* and cleared. The *max_size* and *min_size* are respective upper and lower limits of *M*. Training can only begin after *M* has at least *min_size* experiences. The oldest experience is deleted when *M* has *max_size* elements. In our research, we use a *batch_size* of 16, a *max_size* of 500 000 and a *min_size* of 50 000. We found learning improved when we periodically cleared *M* and refilled it with new experiences every *exp_refill* epochs, where *exp_refill* is 200. The *sim_len* is 4 500 timesteps.

We developed a shallow neural network TSCA to compare against our proposed DQTSCA. The shallow traffic signal

TABLE II: Vehicle Generation Distributions by Flow Rate

Flow Rate (Vehicles/Hour)	**Distribution**	**Parameters** (α, β)
0-150	Inverse Weibull [26]	(0.65, 5.8)
250-450	Burr [27]	(1.4, 5.9)

control agent (STSCA) has one hidden layer with 64 neurons using the sigmoid activation function and four neurons with linear activation functions for its output layer. The state space of the STSCA is two vectors, the first containing elements that represent the number of queued vehicles at each intersection approach (i.e., North, South, East and West) and the second the current traffic signal phase vector *P*. The action space and reward are the same as the DQTSCA. The STSCA is trained using the same number of epochs, action selection policy and gradient descent algorithm as the DQTSCA. However, the traditional agent does not use experience replay, it trains using (2) after every state, action, reward, state sequence.

V. Results and Discussion

The performance of the proposed DQTSCA was assessed with respect to common traffic metrics: throughput, queue length, travel time and cumulative delay. The performance of the agent with respect to the traffic metrics while learning can be seen in Figures 2, 3, 4, and 5. The agent's performance with respect to achieving reward while learning can be seen in Fig. 6. The agent's action-reward performance during one epoch is also shown, in Fig. 7 exclusively exploring initially in training and exclusively exploiting after training in Fig. 8. Initially, while learning, the agent is predominantly exploring (i.e., taking random actions), attempting to learn the action-value function. While exploring, the agent's performance with respect to the traffic metrics exhibits high variance and it achieves negative reward (i.e., punishment). Because of the agent's actions, many vehicles are queued, unnecessarily delayed and the overall throughput is low. As the epochs progress, the agent has better learned the action-value function and can begin selecting exploitative actions instead of exploratory ones. The decreasing exploration rate is reflected in improved performance with respect to all four metrics and higher reward - evidence the agent has learned. Not only does the DQTSCA perform better as training progresses, convergent behavior emerges, as the variance in its performance decreases.

The agent's behavioral change before and after training can be seen in Figures 7 and 8. These figures show rewards as a consequence of each action taken within one epoch (i.e., 1.25 hours of simulated traffic). In Fig. 7, the agent is taking random actions with no consideration for reward, reflected as unstable and divergent rewards. The key observation is that the rewards, positive or negative, increase in magnitude as the epoch progresses because the agent is taking random, exploratory actions. Comparing Fig. 8 with Fig. 7, it is apparent the agent is acting differently. In Fig. 8, the rewards are an order of magnitude smaller and stable as the epoch progresses, with no divergence near the end of the epoch as in Fig. 7, because the agent is enacting an exploitative policy. These observations are supported quantitatively, computing the average and standard deviation (μ, σ) of the reward for each epoch, Fig. 7 has $(-347, 2\,220)$ and Fig. 8 has $(-0.485, 59.6)$.

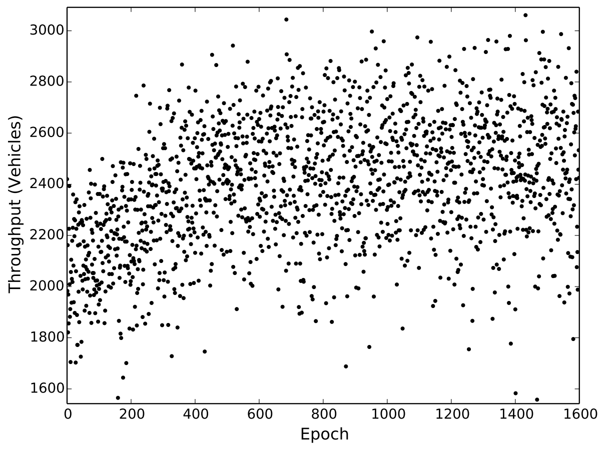

Fig. 2: Intersection throughput while training.

A comparison of the proposed DQTSCA with the STSCA can be seen in Table III. The data in Table III is computed from the last 100 training epochs of each agent, where the agents are taking exploitative action $>93\%$ of the time. Although four traffic metrics are considered, cumulative delay is the only metric the agent can tangibly interact with, as change in cumulative delay is its reward function. The DQTSCA achieves an 82% reduction in the average cumulative delay compared to the STSCA. The difference in this key metric provides evidence that the DQTSCA has learned a control policy superior to the STSCA. Comparing the other traffic metrics, there is no difference in the throughput, but the DQTSCA reduces the average queue length by 66% and the average travel time by 20% compared to the STSCA. The DQTSCA outperforms the STSCA in three of the four metrics, due to the use of the DTSE and its deep architecture. Future work should investigate a throughput reward function and compare the two agents performance, as it is the only metric where the agents perform equally.

A limitation of this research is we did not consider how fair the agent's policy is. A fair traffic signal controller would ensure all vehicles are given equal priority to traverse the intersection, however this may be in conflict with optimizing certain traffic metrics, such as minimization of delay or maximization of throughput. A balance between fairness and optimality could be achieved with the appropriate reward function, which should be the subject of future research.

VI. Conclusion

We proposed, developed and tested a DQTSCA in a traffic microsimulator. The results show deep learning can be applied

TABLE III: STSCA and DQTSCA Traffic Metrics

Traffic Metric (μ, σ, $n = 100$)	**STSCA**	**DQTSCA**
Throughput (Vehicles)	(2 452, 257)	(2 456, 248)
Queue (Vehicles)	(33, 23)	(13, 9)
Travel Time (s)	(197, 107)	(157, 49)
Cumulative Delay (s)	(4 085, 5 289)	(719, 1 048)

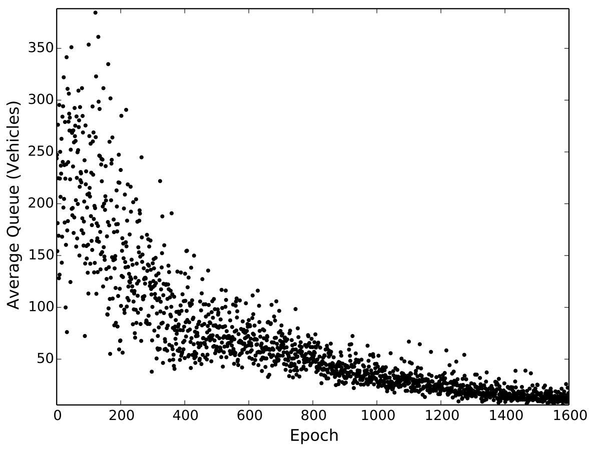

Fig. 3: Average intersection queue while training.

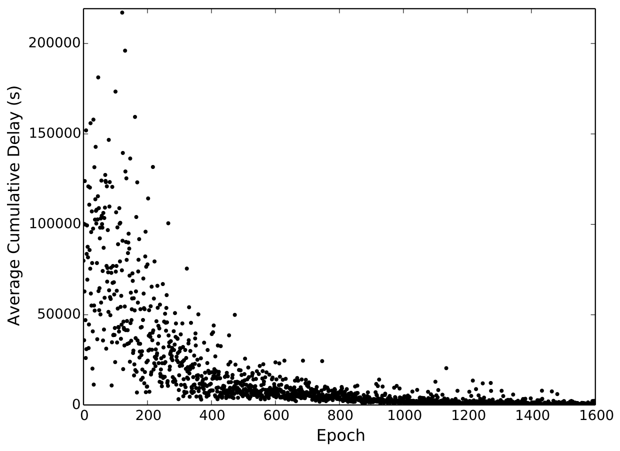

Fig. 5: Average cumulative delay of vehicles while training.

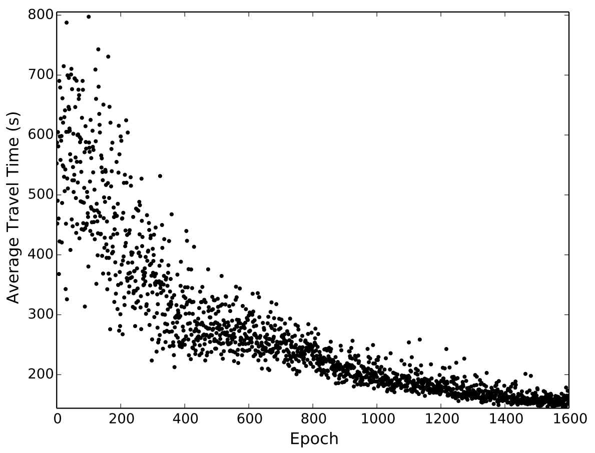

Fig. 4: Average travel time of vehicles while training.

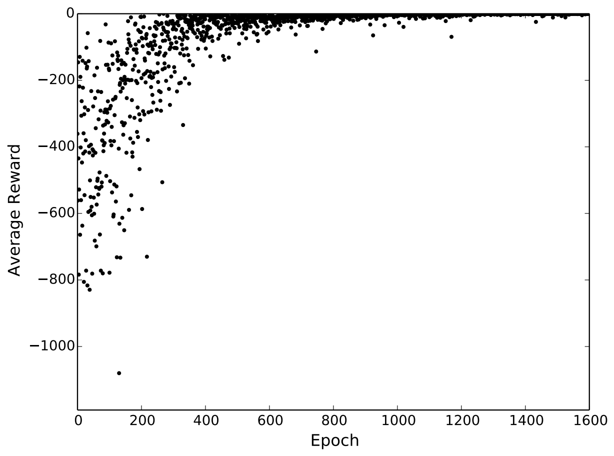

Fig. 6: Average reward of DQTSCA while training.

to traffic signal control with improved performance compared to traditional methods.

Future work in this area can extend the agent's control to all traffic signals, including the yellow and red phases. Currently, the agent has no capability to control the yellow or red phases, they only exist in the transitional sequences between agent actions. However, it is obvious that situations exist where a dynamic yellow or red phase is desirable. For instance, a vehicle does not decelerate to a yellow phase and accelerates through the intersection; extending the yellow phase until all vehicles have either cleared the intersection or are otherwise decelerating is prudent. We hypothesize the means to accomplish this with a TSCA trained through reinforcement learning would be to change the reward function so that it yielded high reward when vehicles decelerate and are not traversing the intersection.

The use of the DTSE may also allow for training a TSCA to control intersections of various lane configurations without retraining. For example, first train the agent using the DTSE on a four lane intersection. Then it could be used to control a two lane intersection by setting all of the elements of the two 'missing' lanes to zero. The DTSE may allow for a widely applicable TSCA without retraining for different intersection geometries.

Additional research should also increase the complexity of the traffic network and apply the DTSE and deep architecture to multiple TSCA. These ideas will be explored in future endeavors by the authors.

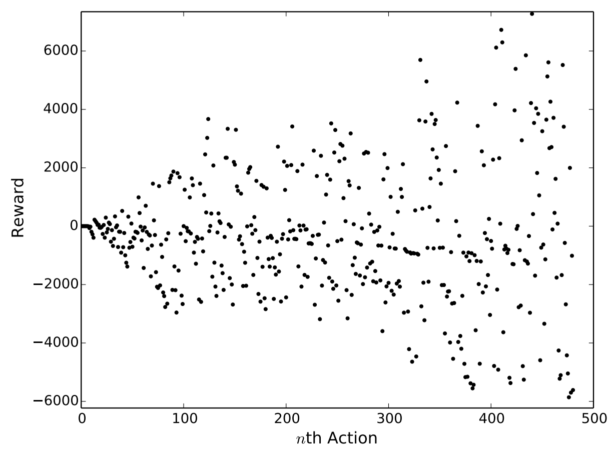

Fig. 7: Reward of DQTSCA in an epoch while taking only exploratory action early in training.

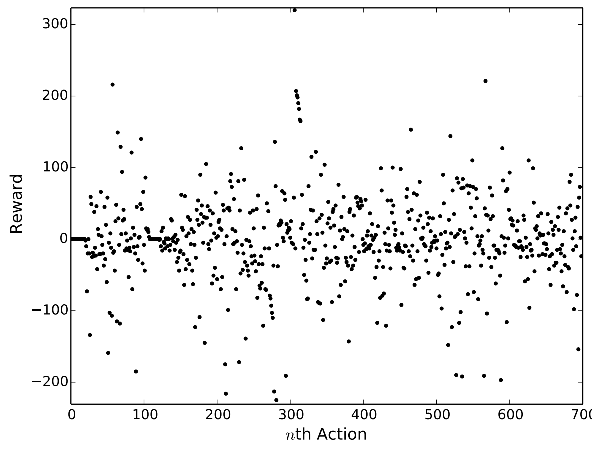

Fig. 8: Reward of DQTSCA in an epoch while taking only exploitative action after training completed.

Acknowledgment

The authors would like to acknowledge the authors of the various programming libraries used in this research. The authors would also like to acknowledge the comments and support of friends, family and colleagues.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[3] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[4] ——, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] P. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[7] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2011.

[8] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[9] T. L. Thorpe and C. W. Anderson, "Traffic light control using sarsa with three state representations," Citeseer, Tech. Rep., 1996.

[10] M. Wiering *et al.*, "Multi-agent reinforcement learning for traffic light control," in *ICML*, 2000, pp. 1151–1158.

[11] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Physical Review E*, vol. 64, no. 5, p. 056132, 2001.

[12] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[13] Y. K. Chin, N. Bolong, A. Kiring, S. S. Yang, and K. T. K. Teo, "Q-learning based traffic optimization in management of signal timing plan," *International Journal of Simulation, Systems, Science & Technology*, vol. 12, no. 3, pp. 29–35, 2011.

[14] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5, pp. 1575–1587, 2013.

[15] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[16] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 227–245, 2014.

[17] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," in *Autonomic Road Transport Support Systems*. Springer, 2016, pp. 47–66.

[18] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[19] C. Hill and G. Krueger, "Its eprimer module 13: Connected vehicles," [Online; accessed 2016-09-07]. [Online]. Available: https://www.pcb.its.dot.gov/eprimer/module13.aspx

[20] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[21] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[22] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[23] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[24] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, may 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[25] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001, [Online; accessed 2016-09-06]. [Online]. Available: http://www.scipy.org/

[26] R. Riccardo and G. Massimiliano, "An empirical analysis of vehicle time headways on rural two-lane two-way roads," *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 865–874, 2012.

[27] A. Maurya, S. DEY, and S. DAS, "Speed and time headway distribution under mixed traffic condition," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 11, no. 0, pp. 1774–1792, 2015.

[28] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[29] J. ONeill, B. Pleydell-Bouverie, D. Dupret, and J. Csicsvari, "Play it again: reactivation of waking experience and memory," *Trends in neurosciences*, vol. 33, no. 5, pp. 220–229, 2010.

[30] L.-J. Lin, “Reinforcement learning for robots using neural networks,” DTIC Document, Tech. Rep., 1993.

Wade Genders earned a Software B.Eng. & Society in 2013 and Civil M.A.Sc. in 2015 from McMaster University. He is currently pursuing a Ph.D. at McMaster University in the Department of Civil Engineering. His research topics include traffic signal control, intelligent transportation systems, machine learning and artificial intelligence.

Saiedeh Razavi is the inaugural Chair in Heavy Construction, and Associate Professor at the Department of Civil Engineering at McMaster University. Dr. Razavi has a multidisciplinary background and considerable experience in collaborating and leading national and international multidisciplinary team-based projects in sensing and data acquisition, sensor technologies, data analytics, data fusion and their applications in safety, productivity, and mobility of transportation, construction, and other systems. She combines several years of industrial experience with academic teaching and research. Her formal education includes degrees in Computer Engineering (B.Sc), Artificial Intelligence (M.Sc) and Civil Engineering (Ph.D.). Her research, funded by Canadian council (NSERC), as well as the ministry of Transportation of Ontario, focuses on connected and automated vehicles, on smart and connected work zones and on computational models for improving safety and productivity of highway construction. Dr. Razavi brings together the private and public sectors with academia for the development of high quality research in smarter mobility, construction and logistics. She has received several awards including McMaster’s Student Union Merit Award for Teaching, the Faculty of Engineering Team Excellent Award, and the Construction Industry Institute best poster award.