Deep Q-learning Approach based on CNN and XGBoost for Traffic Signal Control

Nada Faqir, Chakir Loqman, Jaouad Boumhidi
Department of Computer Science, Faculty of Sciences Dhar El Mehraz
Sidi Mohammed Ben Abdellah University of Fes, Fes, Morocco

***Abstract*—Traffic signal control is a way for reducing traffic jams in urban areas, and to optimize the flow of vehicles by minimizing the total waiting times. Several intelligent methods have been proposed to control the traffic signal. However, these methods use a less efficient road features vector, which can lead to suboptimal controls. The objective of this paper is to propose a deep reinforcement learning approach as the hybrid model that combines the convolutional neural network with eXtreme Gradient Boosting to traffic light optimization. We first introduce the deep convolutional neural network architecture for the best features extraction from all available traffic data and then integrated the extracted features into the eXtreme Gradient Boosting model to improve the prediction accuracy. In our approach; cross-validation grid search was used for the hyper-parameters tuning process during the training of the eXtreme Gradient Boosting model, which will attempt to optimize the traffic signal control. Our system is coupled to a microscopic agent-based simulator (Simulation of Urban MObility). Simulation results show that the proposed approach improves significantly the average waiting time when compared to other well-known traffic signal control algorithms.**

***Keywords*—Convolutional neural network; extreme gradient; traffic control; traffic optimization; urban mobility**

I. Introduction

Congestion causes serious social problems such as long distance travel, fuel consumption, and air pollution. Factors that contribute to traffic congestion include the proliferation of vehicles, poor road infrastructure, and poor signal control. But it doesn't stop people from buying cars, and building new road infrastructure is expensive. A relatively simple solution is to improve the efficiency of traffic light control. Approaches to reduce congestion are still priority topics for researchers from different disciplines. There have been several technological developments to help solve these problems; but there is a real need for further study and analysis of the impact of this problem on the daily lives of millions of people.

According to World Bank statistics released in 2017, 64% of the world's oil is consumed in the transportation sector. The sector also contributes 27% of his CO_2 emissions globally. According to the same source, according to 2022 statistics, domestic and international transport already accounts for 20% of global greenhouse gas emissions. It could rise as much as 60% by 2050 [1].

Artificial intelligence plays a very important role in this topic; agent-based simulations have attained a sufficient degree of complexity and scalability and have shown their ability to manage traffic in the urban environment with other means to improve the quality of life of users.

Some researchers have proposed an optimization method for dynamic routing systems. Although this technique has proven effective in improving computation time, it is rarely used nowadays. Another technique for improving dynamic routing systems is to predict traffic conditions, like in [2] where, a machine learning approach for short-term traffic forecasting was proposed. This approach uses common conditions such as traffic volume, speed, and road segment occupancy to forecast short-term traffic volumes.

Our work presents a contribution to this line of research; and more precisely we will study a simple and always interesting situation: an isolated intersection regulated by traffic lights, which we want to manage by means of an agent capable of exploiting the experience acquired from its learning, possibly representing the admissible traffic conditions. We describe a Convolutional eXtreme Gradient Boosting (ConvXGB) algorithm as a deep reinforcement learning algorithm that automatically extracts all useful functions for adaptive traffic light control from raw real-time traffic data and learns the optimal policy for traffic light control instead of using human-crafted functions. We model the control problem as a reinforcement learning problem. A deep convolutional neural network is then used to extract useful features from the raw real-time traffic data (vehicle positions, velocities, waiting times of each vehicle on the road, traffic light status, etc.) to output the optimal traffic signal control decision. ConvXGB combines the performance of a Convolutional Neural Network (CNN) and eXtreme Gradient Boosting (XGBoost). A key feature of ConvXGB is a systematic strategy for choosing between these two models: XGBoost, widely used by data scientists, is a scalable machine learning system for tree boosting that avoids overfitting. It works well on its own and has proven its prowess in many machine learning competitions. Adding machine learning with CNN, a deep learning class that involves hierarchical learning at multiple different levels, brings clarity.

The simulations are carried out using SUMO (urban mobility simulation tool) of [3] in which we can see the results of the potential regulation of the actions carried out. A considerable reduction in waiting times and vehicle delays is achieved by using the proposed method.

The paper morphology will be as follows: the second section provides a literature research on the applications of Reinforcement Learning (RL) algorithms dedicated to traffic

light control. The third section, we briefly present an explanation of the different techniques used. In the fourth section, we present the studied environment and its concepts. The fifth section describes the structure of the proposed model architecture. The simulation results and performance compared to other reference models are presented in the sixth section and the concluding remarks are given in the last section.

II. Related Work

In general, traffic signals are based on repeated "cycles", where a cycle is a full rotation of all information provided at the intersection, and consisting of a number of phases. According [4] the "phase" is a group of traffic movements through the intersection; undergoing a "green" time interval, followed by "yellow" time and then "red" time. The dedicated time for each phase follows a fixed plan according to [5] method, or by adopting an adaptive green time scenario for each phase according to the traffic dynamics. The RL method applied to adaptive traffic light control has been proven effective in many papers. Multi-agent systems for solving many traffic management and control problems as in [6] research, a challenge in multi-agent environments is whether to approach traffic control as a collaborative problem/domain. In other words, the local optimum usually competes with the global optimum (the entire road network). Dimensional problems related to learning in multi-agent systems. Each implicitly influences the decisions of other agents. Co-evolution and the role of driver behavior in transport networks. The reference [7] introduced Q-learning algorithm for a single isolated intersection, we know that it is a tabular algorithm dealing with a restricted modeling of the environment based on a finite number of states and actions, which only works for this kind of problem and which will suffer from the curse of dimensionality if we ever think of expanding the state space and giving more details on the traffic state. The reference [8], the authors introduced a RL method with context detection for traffic signal control optimization, the detection of the change of context requires the installation of very sophisticated sensors, which is not at all acquired in various road networks throughout the world. The authors of [8] used a variety of RL algorithms to define the impact of the representation of state and action spaces, as well as the choice of actions and the definition of rewards on traffic models for a real-world intersection, and proved the effectiveness of the phasing variable, especially in large and dynamic traffic models. Most of the works deal with the problem of traffic signal control based on human-designed characteristics such as vehicle queue length and average vehicle delay. Human-designed features are abstractions of raw traffic data that ignore useful traffic information and result in suboptimal signal control. For example, vehicle queue length does not take into account vehicles that are not in the queue but are due to arrive soon. This is also useful information for traffic light control. Average vehicle delay reflects only historical traffic data, not real-time traffic demand.

In this work, we draw on the work of [9] and [10] by establishing a more detailed representation of the state space of the studied environment in order to provide more information of the intersection state. In addition, in this paper we propose a new deep learning model for reinforcement learning problems, called "ConvXGB" which is a combination of CNN and XGBoost.

The ConvXGB model does not use human features, nor does it use vehicle queue lengths, instead it automatically extracts all useful features from the raw traffic data. Our algorithm works efficiently at realistic intersections. Our ConvXGB consists of several stacked layers of convolutions that learn the properties of the input and can learn the properties automatically. Then XGBoost predicts the class labels in the last layer. The ConvXGB model is simplified by reducing the number of parameters under suitable conditions, as it does not require readjusting the weight values in the backpropagation cycle.

III. Methodology

A. Deep Reinforcement Learning

The treatment of a problem by reinforcement learning (RL) using Deep Neural Networks (DNN) is called deep reinforcement learning (DRL). RL is a goal-oriented machine-learning algorithm aimed at achieving it by interacting with the environment. For a problem handled by RL, the environment is iteratively observed by the agent (in our case the road network), and it takes an action (by changing the phase or the duration of the phase of the signal). As a result, it receives from the environment a reward (in the case of traffic light control: waiting time …) for the chosen action, the reward will be cumulated with its long-term goal (minimize delay, minimize stops …). Then according to its rules and the state transition probability, the environment switches to the next state.

The RL agent tries to optimize the correspondence between the state space and the action space called policy, by analyzing the rewards discounted, and accumulated in the long run, and from the application of different action sequences. The agent's policy can be adjusted to converge to an optimal policy, by maximizing the expectation of the long-term reward during the learning process. Fig. 1 shows an RL-agent for traffic signal control.

The value function of the reinforcement-learning problem is the estimate of the reward of each pair of state-action in the long run. The value of environment state is the estimated long-run reward discounted by a factor following the policy, as defined in the Bellman equation as follows:

$$v(s) = \sum P(s', r|s, a)[r + \gamma V_{\pi}(s')] \quad (1)$$

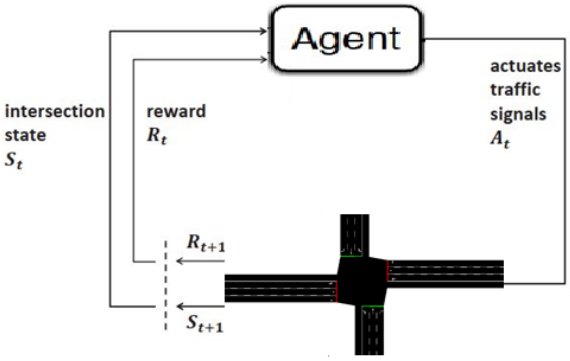


Fig. 1. Reinforcement Learning Agent for Traffic Signal Control.

Where s represent the environment state, a the chosen action, r the reward received from the environment, P the state transition probability, γ the discount factor, π the agent policy and s' the next state according the probability P.

B. Q-learning

Tabular Q-learning according to [11] is a widely used off-policy RL algorithm. Where the chosen action a ϵ A corresponds to the largest Q-value taken from a grid called Q-table. All discrete values of states and actions s ϵ S and a ϵ A will match in this grid. At each time step, Q-learning improves its policy according to the following equation:

$$Q_t(s,a) = (1-\alpha)Q_{t-1}(s,a) + \alpha(r + \gamma\, max_{a' \in A}\, Q_{t-1}(s',a')) \quad (2)$$

At time step t when the agent chooses to execute action a on states it receives a reward r, the parameters α and γ represent respectively the learning rate and the discount factor defining the importance of the upcoming state.

Tabular Q-learning needs a grid to store the Q-values of all pairs (s, a). The learning process suffers from the curse of dimensionality using [11], [12] and [13] approaches, if the sets of states S and actions A increase and if, in addition, the representation of the state is very detailed, so that the grid of Q values becomes giant. Therefore, the function approximation Q(s, a; θ) introduced (where θ is the hyper-parameter of the approximator) aimed at the generalization of the function from an example to estimate the correspondence. In this paper, we used this method by convolutional neural networks while adopting a continuous state representation.

C. Deep Q-Network

The Deep Q-Network (DQN) is a RL algorithm that uses Deep Neural Networks (DNN) as function approximators. Its effectiveness in handling the large state-action space in RL problems as in [14] and in [15] is very significant. In addition, it uses two very important mechanisms to solve instability and divergence problems: experience replay and target network as introduced in [10]. At each learning step, the agent stores the quadruplet (s,a,r,s') in the replay memory, then it takes a random samples as a mini-batch from the memory to update the weights θ. Thus, it eliminates strong correlations between consecutive states. The target network mechanism, which is a neural network identical to the original one, but where weights are updated less frequently, also reduces the correlation problem.

D. XGBoost

XGBoost (Extreme Gradient Boosting) as explained in [16] is a trendy model widely used in several machine-learning challenges. Indeed, it runs faster than other model and is popular for its scalability in all scenarios discussed in [17]. There are many other boosting algorithms such as parallel boosting, regression tree boosting, stochastic gradient boosting, but in [16], the XGBoost model is one of the leading boosting algorithms.

The performance of many machine-learning algorithms depends on their hyper parameter tuning, it is important to tune a hyper-parameter; we used the cross-validation grid search technique to tune the hyper-parameters of XGBoost in this paper. Our results show that this step helped our model achieve impressive scores and consequently beat older methods.

E. Cross-validation Grid Search Tuning Hyper-parameters

XGBoost is a powerful and flexible machine learning algorithm and also it performs very well in general, but there are some problems. Notably the large number of hyper-parameters it has, as well as the fact that different combinations of parameters generate different evaluation scores. Hence it is essential to find the optimal hyper-parameters to get the most out of it. Grid search is method to find optimal hyper-parameters by testing every combination of them. In our case, we choose to tune three common parameters to prevent over fitting: learning rate, minimum child weight and maximum depth.

F. ConvXGB Model

As a fresh deep learning model for categorization issues, we introduce the Convolutional eXtreme Gradient Boosting (ConvXGB) technique. ConvXGB combines the strengths of a Convolutional Neural Network (CNN) with eXtreme Gradient Boosting (XGBoost) [16], resulting in a state-of-the-art performance and high accuracy, as we will demonstrate. The ConvXGB architecture consists of three sub-CNN-networks each containing three stacked convolutional layers, the outputs of which are merged and reshaped, thus serving as the input to the XGBoost which is the final layer of the model. They differ from conventional CNN in that there is neither a pooling couch nor a fully connected (FC) couch. This adds simplicity and reduces the number of calculation parameters because it is not necessary to adjust the weight in the FC couches in order to adjust the weight in the preceding couches.

IV. Description of the Reinforcement Learning Environment for Traffic Lights Control

A. Intersection Model

The environment, on which the agent operates, is a four-armed intersection shown in Fig. 2. Each arm is 500 meters long, the maximum speed is 70km/h and the adopted traffic scenario handles an average of 1000 vehicles per hour. There are three lanes on each arm defining the possible directions of a vehicle: the rightmost lane allows vehicles to turn strictly right, the middle lane allows the driver to go straight only, while on the leftmost lane the driver can turn left or make a U-turn.

In the center of the intersection, the agent controls an adaptive traffic light system. Pedestrians, pavements and pedestrian crossings are not included in our environment.

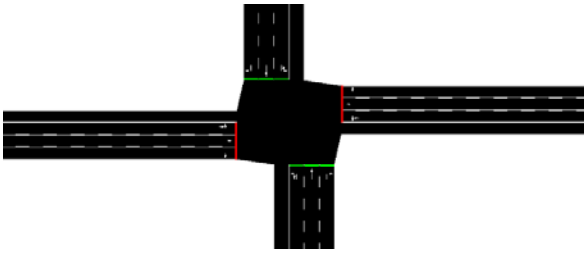


Fig. 2. Intersection Model.

B. State Representation

A detailed representation of the state of the environment is provided for the RL agent. The state is provided in the form of three matrices on each incoming road: a matrix of the vehicle's position and the matrix of the vehicle's speed used in [10], [18], [19] and [20]; and a waiting time matrix of the vehicle's, and also the last traffic lights status [21]. To create the three matrices P_i, V_i and W_i, we will need to consider two parameters the segment length l and the cell length c for each road i = 0, 1, 2, 3; Fig. 3.

In this paper hot coding has been used to define the position matrix of the road i, the result is the matrix P_i. The speed matrix V_i is calculated by normalizing each vehicle's speed by the edge maximum allocated speed. In our case, we consider the waiting time from the moment the vehicle enters the incoming route i of the network, and the result is recorded at the corresponding entry in the matrix Wi. The waiting time of a vehicle is defined as the amount of time during which a vehicle has been in the "waiting" state.

From the state of the last activated traffic lights, a matrix L is generated, where: L = [1, 0] is a value that defines green lights on horizontal roads whereas L = [0, 1] represents green lights on vertical roads as used in [10]. In this way, the state representation of the intersection at each time step t provided to the RL agent will be S_t = (P, V, W, L) ∈ S, where S is the entire state space.

C. Action Definition

In Fig. 4, at each time step the agent observes the state of the intersection then chooses and executes either action (A_t = 0): "0" setting green lights on for horizontal roads or action(A_t = 1): "1" setting green lights on for vertical roads. A transition phase is required if the action taken at time step t is different from the one chosen at time step t+1, it will be executed as follows:

1) Switching the lights for straight ahead vehicles to yellow.

2) Switching the lights for straight ahead vehicles to red.

3) Switching the lights for left-turning vehicles to yellow.

4) Switching the traffic lights for vehicles turning left to red.

The times allocated for the yellow and green lights are fixed and are worth six seconds and ten seconds respectively.

D. Reward Definition

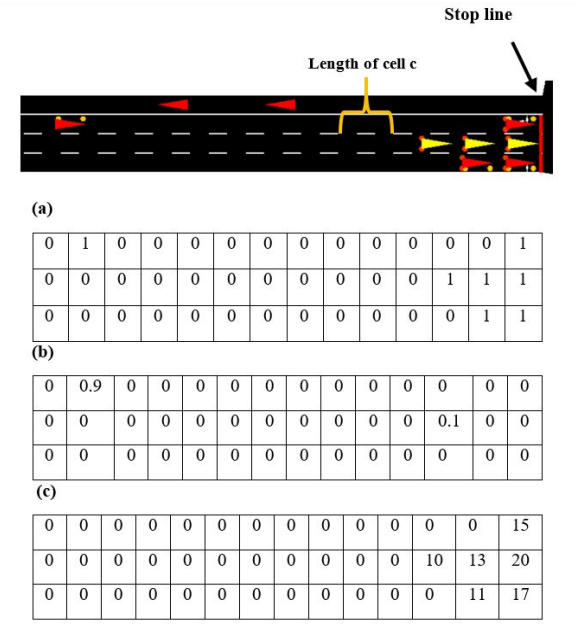
In reinforcement learning, the feedback that the agent receives from the environment as a result of his choice of action is called a reward. This crucial concept of the training process is used to measure the effectiveness of the choice of action and thus allows the RL agent to improve future choices of his actions.

The reward can generally take positive or negative values. The positive value is the result of the right choice of actions; whereas a negative value is due to the wrong choice of actions. In our case, the objective is to minimize the total waiting times of all vehicles in the intersection. So that the RL agent can measures the effect of the action taken on the effectiveness of the adaptive control of traffic lights at the intersection in terms of reduction or increase, the reward must be derived from a performance measure of traffic efficiency, so the goal will be met. The intersection is observed for two times, once at the beginning of the time step and then at the end of the time step.

The waiting times in the arrival lanes are noted for each observation and for all vehicles present in the intersection. The formula for the total reward R_t also used in [10] is therefore:

$$R_t = r_1 - r_2 \quad (3)$$

The values of r_1 and r_2 are respectively the sum of the waiting times recorded at the beginning and at the end of the green light interval according to [10]. The agent will only be rewarded if the value of r_2 decreases; in case a transition phase is mandatory ($A_t \neq A_{t+1}$), the two values r_1 and r_2 will be saved at the end of the execution of the transition phase discussed in section IV.C.



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

0	0.9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	15
0	0	0	0	0	0	0	0	0	0	0	10	13	20
0	0	0	0	0	0	0	0	0	0	0	0	11	17

Fig. 3. Illustration of One Arm of the Intersection and the Three Matrices Position (a), Speed (b) and Waiting Time (c) of each Vehicle in the Incoming Lane.

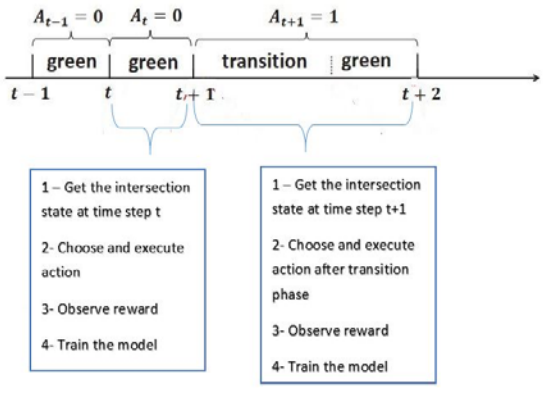


Fig. 4. Chronological Sequence of Events of the Agent: State Observation, Action Choosing, Executing, Getting Rewards and Training Model According.

V. Adaptative Signal Control Algorithm Design based on ConvXGB

The ConvXGB agent, which is a RL agent, long-term objective is to reduce congestion at the intersection. After making a series of decisions (actions) in accordance with a policy, we consider S_t, the intersection's status at the start of time step t, into consideration. A sequence of rewards R_t, R_{t+1}, R_{t+2}, R_{t+3}... is given to the agent, after taking a sequence of actions by adopting a policy π for a state S_t of the environment at the beginning of each time step t. The action A_t should maximize the reward R_t shown in equation (4) to allow the agent to reduce the total waiting times of all vehicles at the intersection at time step t. The agent is supposed to find an optimal policy for choosing actions, denoted π^* defined in (5) that maximizes the cumulative future reward (Q-values) and that helps it achieve its goal.

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

$$\pi^* argmax_\pi Q_\pi(s,a) \; for \; all \; s \in S, a \in A \quad (5)$$

To avoid infinite returns in cyclical processes, rewards must be discounted. Thus at each time step the reward is weighted by a reduction (or discounting) factor $\gamma \in [0, 1]$. This discount factor means that the further into the future one is, the less important the rewards become. We denote the optimal values of Q under policy π^* as $Q^*(s,a) = Q_{\pi^*}(s,a)$. Our model extract useful traffic features, and predict the Q-values while trying to find the optimal policy π^* and thus find the optimal Q-values $Q^*(s,a)$. Bellman's optimality equation (6), gives a recursive relation for the optimal Q-values $Q^*(s,a)$.

$$Q^*(s,a) = E[R_t + \gamma Q^*(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \quad (6)$$

In order to calculate (6), we would need to have complete knowledge of the underlying system mode (state transition probabilities and associated rewards), which is actually not feasible. Due to the complexity and size of the intersection traffic space, ConvXGB model is used to approximatively determine the optimal Q-values. It receives as input the state representation St = (P, V, W, L). Based on this traffic data, the CNN sub-networks extract useful features, which they then input to XGBoost to forecast the estimated $Q(S_t, a)$ for all actions $a \in A$ given the observed condition S_t.

A. Proposed ConvXGB Architecture

Our paper is based on a new deep learning model, dedicated to reinforcement learning problems called "ConvXGB". It is a hybridization between convolutional neural network and an XGBoost model. The architecture of ConvXGB is shown in Fig. 5. The model has five layers: 1) input layer, 2) three identical CNN sub-networks each containing three stacked convolutional layers, 3) reshape layer, 4) Q-values prediction layer and 5) output layer. These layers, each of which has unique talents and duties, are crucial to the model's success. Further, the model can be divided into two parts: one for feature learning and the other for Q-values prediction.

These three identical sub-networks having respectively as inputs the matrices (P, V, W). The outputs of these three sub-networks is flattened, and then we merge them with the vector L (last state of the traffic lights) to provide the result to the third layer for reshaping, Table I shows all the CNN parameters.

The second part of the ConvXGB model is an XGBoost model, with tuned parameters using GridSearchCV technique, to predict the Q-values. XGBoost are optimized by a cross-validation grid search.

The performance of the hybrid ConvXGB model is compared to the classical DQN model using only convolutional networks to show its robustness. ConvXGB effectively uses features learning and predicts Q-values, thus significantly reducing the total waiting times for all vehicles in the intersection, which will be detailed in the results section.

TABLE I. Parameters of the CNN Layers

CNN Layers	Number of filters	Stride	Activation function
Conv1	16 of 4x4	2	ReLU
Conv2	32 of 2x2	1	ReLU
Conv3	64 of 2x2	1	ReLU

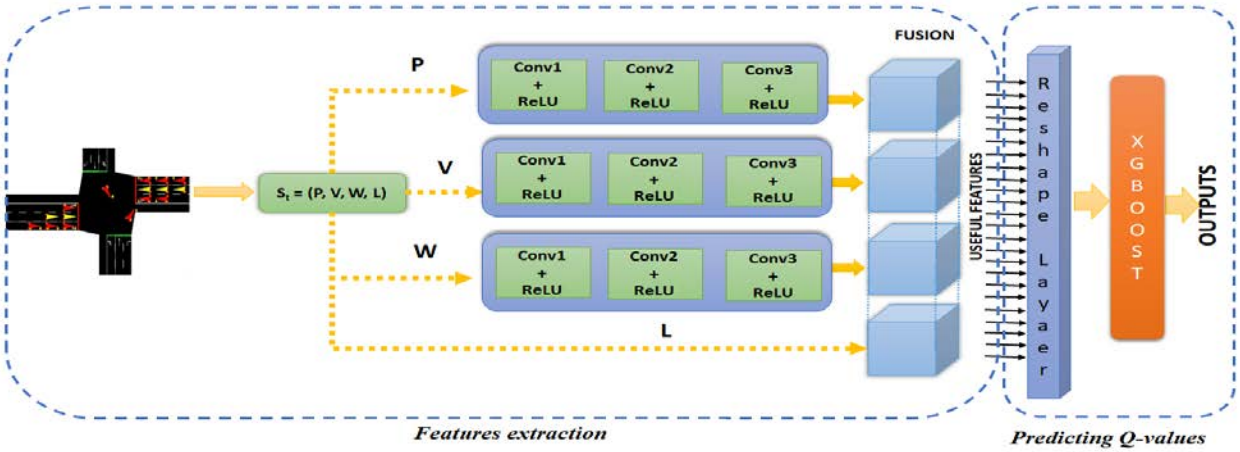


Fig. 5. Architecture of the ConvXGB Model.

B. Algorithm and Training Process

The algorithm in Fig. 6 summarizes the process of forming the ConvXGB model used. We use a hybrid model composed of three identical CNN sub-networks to extract useful features from an environment representation (state). In the prediction phase, we use XGBoost to make Q-values predictions given the features extracted by CNN sub-networks. We also used the experience replay used in [10] to break correlations and improve agent performance during the training phase. The replay memory provides to the RL agent a set of random experiences for its learning, called a mini-batch. Each experience is represented as a quadruplet { S_t, A_t, R_{t+1}, S_{t+1}}, collected and stored in memory M during the learning phase, after choosing and applying the action A_t from the state S_t, the agent receive R_{t+1} as reward.

The environment undergoes a transition from the state S_t to the state S_{t+1}. This is why the observations extracted from this environment are correlated. This correlation can limit the learning capacity of the RL agent and prove the need to use the replay memory. The oldest experiments are erased if the memory is full during learning. The agent needs training data: input data set X = {(S_t, A_t): t ≥ 1} and the corresponding targets y = {$Q^*(S_t, A_t)$: t ≥ 1}. For input data set, (S_t, A_t) can be retrieved from replay memory M. However, target $Q^*(S_t, A_t)$ is not known. This formula $R_t + \gamma\, max_{a'}\, Q\,(S_{t+1}, a')$ is used to estimate the optimal Q-values. Thus, targets y = { $R_t + \gamma\, max_{a'}\, Q\,(S_{t+1}, a')$: t ≥ 1}.

The RL agent randomly takes 32 quadruples ({ S_t, A_t, R_{t+1}, S_{t+1}}) of the replay memory used to form training data X = {(S_t, A_t): t ≥ 1} and also to tune and update XGBoost parameters using GridSearchCV technique to efficiently estimate optimal Q-values (Q^*).

Algorithm 1 ConvXGB for Adaptative traffic signal control using Q-learning

```
Initialize replay memory M
Set the parameters of the CNN sub-networks for learning features
Initialize XGBoost model parameters model with random parameters for the prediction step
Initialize ϵ, β, γ, episodes
for episode = 1 to episodes do
    Start new time step
    while step ≤ maxsteps do
        Get current State observation S_t
        The CNN sub − networks extract useful features from S_t
        The XGBoost model select A_t = argmax_a Q(S_t, a) with probability 1 − ϵ and a random action with probability ϵ
        if A_t = A_{t−1} then
            No transition phase
        else
            Make a transition phase for traffic signals
        end if
        Execute selected action
        Increment simulation time step
        if time step ends then
            Observe reward R_t and current state S_{t+1}
            Store quadruple (S_t, A_t, R_t, S_{t+1}) into M
            Randomly draw mini − batch of 32 samples from M
            Delete M[0] if the memory is full
            Form training data using data set X as inputs and y as targets
            Tune XGBoost parameters on training data X and y using GridSearchCV technique
            Update XGBoost model with optimal parameters to predict Q values
        end if
    end while
end for
```

Fig. 6. ConvXGB Algorithm for Adaptative Traffic Light Control using Q-Learning.

The exploration / exploitation problem is a frequent problem facing the policy of choice of actions in reinforcement learning; exploration, where we seek more information to improve future decisions; or exploitation, where the decision is made based on current information. In this paper, we have adopted a ε-greedy exploration policy as used in [18]; given in equation (7). For the current episode **e** we have the probability **ε** of an exploratory action, and the probability **1 - ε** of an exploiting action.

$$\varepsilon_e = 1 - \frac{e}{Total\_episodes} \tag{7}$$

At the start of his training, the agent only explores his environment, which is logical and obvious; he begins to use the information received during his training; an exclusive exploitation of the knowledge acquired by the agent takes place towards the end of his training.

VI. Experiment and Evaluation

A. Simulation Settings

In this study, we used the SUMO simulator [1], to generate urban traffic simulations. It allowed us to implement and customize road infrastructure functionalities. Moreover, subsequently extract useful data during the traffic simulation.

1) Intersection: We have an intersection of four roads, each road having three lanes, as shown in Fig. 2. The length of the road is 500 meters, the maximum speed is 19.44 m/s (i.e. 70 km/h), and the length of the vehicles is 5 meters with a minimum distance between vehicles of 2.5 meters. Moreover, the flow of vehicles (an average of 1000 per hour) is uniformly distributed.

2) Traffic: Vehicles, while selecting their route in advance make the choice of entry routes randomly. Horizontal roads will be heavily used while vertical roads, which will be less frequented, we also increase the frequency for left deviating roads compared to right deviating roads. Specially, P_1 = 1/7 (for horizontal roads), P_2 = 1/11 (for vertical roads), P_3 = 1/30 (for roads deviating to the right), P_4 = 1/25 (for roads deviating to the left). The setting of the traffic lights and the transition phase used in this paper are detailed in Section IV.

3) Agent parameters: The training process takes place over 2000 episodes. Where each episode corresponds to two hours of traffic and 7000 time steps, simulating the same traffic load at peak hours. For ε-greedy method in Algorithm 1, parameter ε is set to be 1 and the discount factor γ = 0.95. Learning rate of RMSProp α is set to 0.0002 and the replay memory can store the experiences of 200 episodes. The training of the hybrid agent took a few days non-stop high-end laptop.

4) Simulation data: The time (in seconds) that a vehicle takes to cross an intersection (between entering and exiting the lanes) defines the delay that it can make. Therefore, the waiting (stopping) time of a vehicle is closely related to its delay. During the simulations, the total waiting time of all vehicles at the intersection is recorded in a file for all episodes.

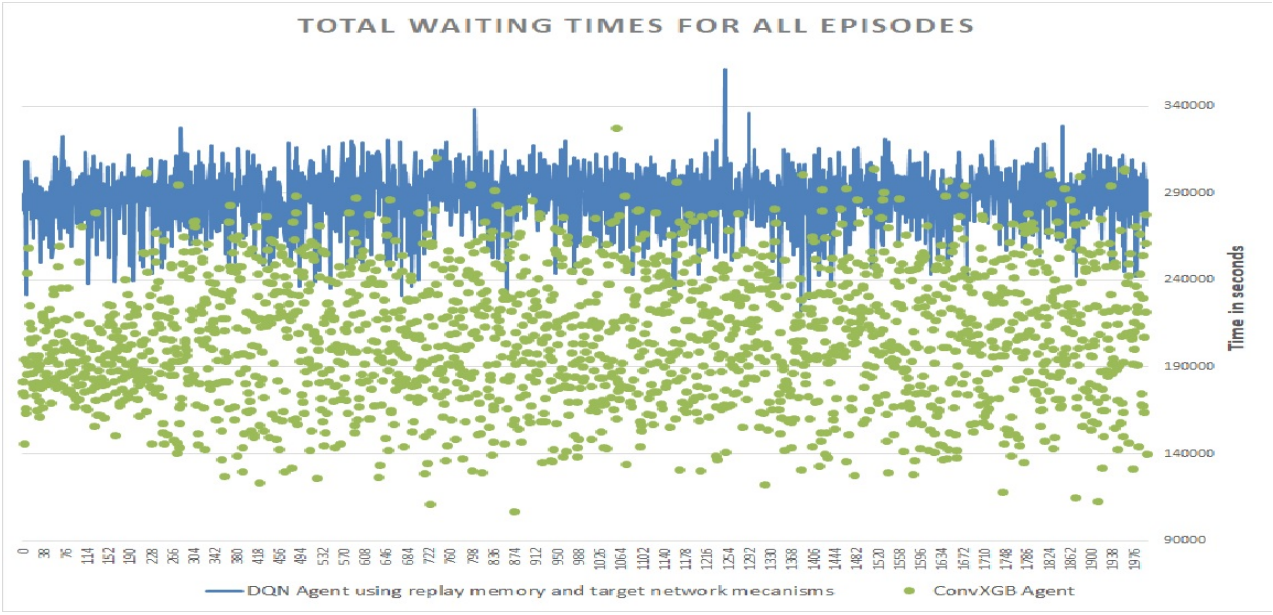


Fig. 7. ConvXGB Agent Improvement.

Our approach is based on a hybrid model as detailed in Section V according to an architecture defined in Fig. 6 and inputs (P, V, W, L) explained in Section IV. The results of our contribution using used a tuned ConvXGB approach, shown in Table II, are compared to the fixed time policy, in which the phase cycle is fixed throughout the day, however the timing of each phase is designed by an expert to accommodate traffic volume ratios and to prioritize arterial traffic for green-waves, and also compared to the approach of [10] that we simulated for our intersection. Approach of [10] showed a clear improvement over the fixed traffic light configuration, but the hybrid ConvXGB agent we developed was able to outperform the fixed time policy and the mentioned approach very well. Fig. 7 presents the performance of the mentioned methods.

Using our agent, we obtain approximately 192k seconds as average waiting times for all episodes, while for the agent in [10] it was approximately 285k seconds. This proves that our ConvXGB hybrid model with XGBoost hyper-parameter tuning has significantly improved the problem of reducing traffic congestion.

TABLE II. TOTAL WAITING TIMES OF ALL VEHICLES FOR THE IMPLEMENTED APPROACHES

	Mean of total WT of all vehicles in seconds	**Best total WT of all vehicles in seconds**	**Improvement over the static scenario**
DQN agent [10]	285503	231444	54%
ConvXGB agent	**192213**	**145425**	**72%**
Fixed time policy	505850		

B. Training Evaluation Results and Discussion

The problem of this study was as follows: can a hybrid model "Convolutional Neural Network-Extreme Gradient Boosting" statistically outperform the "DQN" model in optimizing the control of traffic lights in the urban environment?

The quantitative and complete description of the traffic situation provided to the agent helped a lot. It is clear that the states are more complex but the agent gains in performance. The inputs of our model were designed in a suitable way, in contrast to [22] where the authors used binary position matrices. They defined the binary matrix to cover the entire rectangular area around the intersection instead of just covering the area of the street relevant to traffic light control. Most entries in the binary matrix are null and redundant, making the binary matrix inefficient because the vehicle can't travel on non-road areas. In our case the vehicle position matrix covers only intersecting roads. This reduces the cost of training computation. We were able to boost the performance of our ConvXGB model by using the GridSearchCV technique for fine-tuning the XGBoost model. And since the tests we were able to do this step really helped our model to show better results.

Based on the evaluation and comparison of the two models, the results clearly showed the performance difference of the ConvXGB hybrid model. As shown in Fig. 7, the results obtained are significantly larger than the approach of [10]. The ConvXGB-based Q-learning algorithm is an effective choice over traditional traffic control methods, solving the problem of traffic congestion in large cities.

In the future, a better prediction model could be developed by using a heuristics methods for fine-tuning of the XGBoost and experimenting with new hybrid algorithms is recommended for future work.

VII. CONCLUSIONS

We have developed a new deep learning model for traffic light control problems using reinforcement learning. ConvXGB has two parts: one for useful feature extraction based on

detailed traffic situation definition and one for predicting Q values according to Q learning algorithm. We evaluated ConvXGB, on a single intersection, by adopting a traffic scenario illustrating clear peak hour congestion. ConvXGB was simplified by reducing the number of parameters needed and did not require back-propagation in the fully connected layer. ConvXGB based on CNN and XGBoost, was improved by tuning the most common XGBoost hyper-parameters. Our experimental results show that our model is significantly better than the classic DQN model, which also performed well when comparing it to the fixed time policy, but its performance is clearly inferior to our ConvXGB.

REFERENCES

[1] "Transport overview." [Online]. Available: http://www.worldbank.org/en/topic/transport/overview#1. [Accessed: 23-sep-2022].

[2] O. Mohammed and J. Kianfar, "A Machine Learning Approach to Short-Term Traffic Flow Prediction: A Case Study of Interstate 64 in Missouri," *2018 IEEE Int. Smart Cities Conf. ISC2 2018*, pp. 1–7, 2019, doi: 10.1109/ISC2.2018.8656924.

[3] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO--simulation of urban mobility: an overview," Proc. SIMUL 2011, Third Int. Conf. Adv. Syst. Simul., 2011.

[4] S. Touhbi et al., "Adaptive Traffic Signal Control: Exploring Reward Definition for Reinforcement Learning," Procedia Comput. Sci., vol. 109, pp. 513–520, 2017, doi: 10.1016/j.procs.2017.05.327.

[5] Webster, "Traffic Signals Webster," Road Res. Tech. Pap., no. 39, pp. 1–44, 1958.

[6] A. L. C. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," Auton. Agent. Multi. Agent. Syst., vol. 18, no. 3, pp. 342–375, 2009, doi: 10.1007/s10458-008-9062-9.

[7] B. Abdulhai and L. Kattan, "Reinforcement learning: Introduction to theory and potential for transport applications," Can. J. Civ. Eng., vol. 30, no. 6, pp. 981–991, 2003, doi: 10.1139/l03-014.

[8] D. De Oliveira et al., "Reinforcement learning-based control of traffic lights in non-stationary environments: A case study in a microscopic simulator," CEUR Workshop Proc., vol. 223, 2006.

[9] S. El-Tantawy and B. Abdulhai, "Comprehensive analysis of reinforcement learning methods and parameters for adaptive traffic signal control,", Proc: Transportation Research Board 90th Annual Meeting, Washington DC, USA, Jan. 23-27-2011.

[10] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network," pp. 1–10, 2017.

[11] C. Watkins and P. Dayan, "Q-learning,", Machine learning, vol.8, pp. 279–292, 1992, doi:10.1007/BF00992698.

[12] C. El Hatri and J. Boumhidi, "Traffic management model for vehicle re-routing and traffic light control based on multi-objective particle swarm optimization,", Intelligent Decision Technologies, vol.11, no.2, pp. 99-208, 2017, doi:10.3233/IDT-170288.

[13] M. Tahifa, J. Boumhidi, and A. Yahyaouy, "Multi-agent reinforcement learning-based approach for controlling signals through adaptation," Int. J. Auton. Adapt. Commun. Syst., vol. 11, no. 2, pp. 129–143, 2018, doi: 10.1504/IJAACS.2018.092019.

[14] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[15] S. S. Mousavi, M. Schukat, and E. Howley, "Deep Reinforcement Learning: An Overview," Lect. Notes Networks Syst., vol. 16, pp. 426–440, 2018, doi: 10.1007/978-3-319-56991-8_32.

[16] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. 13-17-Augu, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.

[17] C. Bentéjac, A. Csörgö and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms,", Artificial Intelligence Review, vol. 54, pp. 1937–1967 , 2021, doi: 10.1007/s10462-020-09896-5.

[18] W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," Procedia Comput. Sci., vol. 130, no. July, pp. 26–33, 2018, doi: 10.1016/j.procs.2018.04.008.

[19] W. Genders and S. Razavi, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control," , ArXiv, vol. abs/1611.01142, pp. 1–9, 2016.

[20] K. Shingate, J. Komal and D. Yohann,, "Adaptive Traffic Control System using Reinforcement Learning," Int. J. Eng. Res., vol. V9, no. 02, pp. 443–447, 2020, doi: 10.17577/ijertv9is020159.

[21] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A Survey on Traffic Signal Control Methods," , arXiv learning ,vol. 1, no. 1, 2019.

[22] E. Van Der Pol, "Deep Reinforcement Learning for Coordination in Traffic Light Control," *Master thesis*, no. November 2015, p. 2016, 2016.