How to Train your ITS? Integrating Machine Learning with Vehicular Network Simulation

Max Schettler*, Dominik S. Buse*†, Anatolij Zubow*, and Falko Dressler*
*School of Electrical Engineering and Computer Science, TU Berlin, Germany
†Software Innovation Lab, Paderborn University, Germany
{schettler,buse,zubow,dressler}@ccs-labs.org

***Abstract*—Machine Learning (ML) is becoming ever more popular in many application domains, including vehicular networking. It has been shown already that Intelligent Transportation Systems (ITS) can greatly benefit from this approach, particularly from Reinforcement Learning (RL). To implement Vehicular Adhoc Network (VANET) environments for RL training, researchers often start from scratch. Because up until now, there is neither an established interface to ML toolkits nor a common scenario for VANET applications. Though such established standards would be a great benefit to research: Previous results would be easier to reproduce and different solutions could be compared in equal situations and using the same metrics. We developed `Veins-Gym` to bridge this gap. `Veins-Gym` combines the popular Veins vehicular networking simulator with OpenAI Gym. Using an exemplary VANET application, we show that RL techniques can be easily applied to ITSs with this framework. This enabled us to train an agent that outperformed hand-written algorithms.**

I. Introduction

Intelligent Transportation Systems (ITS) have the potential to take road traffic to the next level. Based on Vehicular Adhoc Network (VANET) technology, they enable new ways to enhance traffic safety, efficiency, and convenience [1]. VANETs, due to their distributed nature, are inherently complex and hard to configure. Especially so, as the individual elements of the network need to behave correctly so the system as a whole can work: Entire stacks of protocols have been developed and eventually standardized. However, each protocol in theses stacks has many configuration parameters, which play an important role in the performance of the protocol, and the interplay among them. As the environment of the network may change rapidly due to mobility of the network, there is no generally valid optimal set of parameters.

With the rise of Machine Learning (ML) technology, there is an alternative to the manual search for protocol configurations [2]. Deep Neural Networks (DNNs) can be trained to yield configuration by examples. They can be trained using observations of relevant situations and known desired outcomes to become able to yield desired outputs for new observations. The field of Reinforcement Learning (RL) takes this idea another step further [3]–[5]. It is no longer necessary to produce possibly large sets of training data with known desired outcomes before training can start. Instead, only some *reward* function is necessary that judges how good a particular behavior is, given a particular state of the environment. Then, an autonomous *agent* will explore its environment to gather observations, produce behavior based on prior training (or simply guessing), and adapt according to said reward function.

In the RL community, the concept of a *gym* has gained much traction after being proposed by Brockman et al. [6] for the OpenAI project. A gym provides a standardized interface to an environment suitable for an RL agent, reducing entry barriers and necessary boilerplate, and enabling interfacing to other projects.They have been successfully adapted in domains such as robotics [3], road traffic control [4], or wireless networks [5]. But, to the best of our knowledge, there is no established gym for VANET research yet.

VANET research often uses tools such as Veins [7] to simulate vehicle mobility, wireless transmissions, and protocol behavior. Veins provides protocols, models, and coupling to a mobility simulator, which enables simulation of large scale vehicular networks. Since it is implemented inherently deterministic, it can be used to conduct reproducible experiments.

We aim at bridging the gap between VANET research and ML. By coupling Veins to OpenAI Gym, we combine these two domains. This enables researches familiar with either domain to apply their knowledge in the other scope: VANET researchers gain an interface compatible with common RL frameworks that they can use to enhance their protocols. ML researchers on the other hand may chose to investigate the performance of a particular learner on a VANET problem. Using the new `Veins-Gym` framework, we provide an example of how learning techniques can be applied in VANET research.

Our key contributions can be summarized as follows:

- We developed `Veins-Gym`, an integration between Veins and OpenAI Gym, which allows to easily formulate and test ML/RL solutions in the domain of ITS, we will shortly release the framework as Open Source;[1]
- we apply RL to solve an exemplary VANET problem; and
- we train an agent that solves the problem exceeding the performance of both naïve and self implemented strategies.

II. Related Work

The concept of a gym for ML training was originally introduced in [6]. Since then, gyms have been used and extended for a variety of topics. Traditionally, they considered domains such as control theory and robotics [8], which make up the majority of the gym environments in the OpenAI registry.[2]

[1] http://www.tkn.tu-berlin.de/software/veins-gym/
[2] https://gym.openai.com/envs

The *flow project* [4] gained traction by coupling the SUMO traffic simulator to a gym environment. It enables research of road traffic control and aims to provide a benchmarks for relevant traffic situations that can be handled using RL agents. Even though it considers connected vehicles, there is no simulation of wireless communication in these environments.

The *ns3-gym* [5], on the other hand, provides a framework to simulate communication network environments using the *ns3* network simulator. It simplifies and codifies the creation of environments for agents controlling the behavior of network protocols. However, it lacks the simulation of vehicular mobility required for VANET research.

Choe et al. [9] tried to tackle this issue by additionally coupling SUMO to the *ns3-gym*. The paper focuses on the optimization of Medium Access Control (MAC) protocols – specifically such based on IEEE 802.11p – using the resulting coupled environment. Pressas et al. [10] undertook similar approaches to enhance vehicular MAC protocols using RL. They evaluated their own agents using a combination of SUMO and the OMNeT++ simulation stack. Thus, other ML algorithms cannot easily be integrated.

In conclusion, ML and particularly Reinforcement Learning is being actively investigated in the VANET domain. However, the entry barriers are still high as researches have to reinvent the wheel and combine RL toolkits and VANET simulators. A shared gym environment for VANET scenarios using the well-known Veins simulator could ease these issues.

III. Gyms for VANET Research

We developed `Veins-Gym` to make the vehicular network simulator Veins accessible for use with ML/RL frameworks. Similar to *ns3-gym* [5], `Veins-Gym` is a framework that provides the basic connection between the gym interface and the simulator. It further simplifies the process of modeling specific VANET scenarios into environments suitable for RL agent training.

An OpenAI gym provides an agent with a standardized interface to the environment, encapsulated in a single Python class. There are two major points of interaction with the environment, each implemented as a method of said gym class. With *reset*, the environment is (re-)started and put into an initial valid state. It returns the *observations* that describe environment to the agent. In future steps, additionally a *reward* value is given that judges the outcome of the last action and can be used to train the agent. Furthermore, a *done* flag is returned that indicates whether the environment is considered completed and needs to be restarted. The agent can then choose how to react to these observations and call the *action* method. This performs the chosen action in the environment, advances the simulation, and returns a tuple of *observations*, *reward*, and *done*-flag describing the updated state. The agent usually executes the *action* method in a loop, reacting to the changing environment, until the scenario is done. Aside from those main interaction points, methods exist to set the seed for Pseudo-Random Number Generators (PRNGs) used by the environment, or to render its state, if applicable.

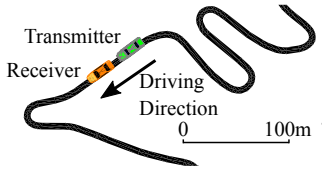

Figure 1. The two vehicles driving along the serpentine road.

In order to couple Veins with an RL agent, we developed `Veins-Gym` in form of a Python class that implements this interface. It manages the life cycle of an embedded Veins process and communicates with it. When constructing the environment, additional simulation parameters are gathered that can be used to override defaults, which is useful during development, or for hyper-parameter studies. The concrete encoding of observations, actions, and reward depend on the problem modeled by each gym environment. `Veins-Gym`, in accordance with the OpenAI gym concept, is agnostic to the concrete encoding and can thus be used to model various VANET scenarios and problems. It simply provides a way to specify these encodings and transport them from the gym class to the simulation and back.

To define a specific problem, the gym environment needs to be adapted in two places: the Python gym class and the C++ code of the Veins simulation. Both need to share a compatible specification of the desired observation, action, and reward space. In the Veins simulation code, a connector class needs to be instantiated to represent the agent in the environment. Whenever the agent has to decide on an action, the observation, reward, and done information have to be passed to the gym interface using said connector. In turn, the connector receives the encoded action chosen by the agent, which is then decoded and executed by the code in the simulation. In practice, the change to a given VANET scenario is relatively small and changes can be quickly implemented.

IV. A Learnable VANET Problem

In order to evaluate the feasibility of Reinforcement Learning for VANET problems, we designed an example environment: Two vehicles drive behind each other along a serpentine road (cf. Figure 1). The vehicle in the back periodically tries to transmit messages to the vehicle in front. Both vehicles are equipped with two different communication methods that have different properties: Dedicated Short Range Communication (DSRC), i.e., IEEE 802.11p, and Visible Light Communication (VLC)-enabled head- and taillights. The agent's goal is to select the optimal communication link for each message. As the road in the scenario contains many sharp turns, the VLC link will not work at all times. But the DSRC link is considered expensive due to its large interference domain. So the optimal link technology depends on the current relative location of the two vehicles.

A. Problem Definition

The two vehicles drive along an except of the Lysevegen pass in southern Norway and 7750 m long in our scenario. It

has a total of 28 hairpin turns, providing plenty of challenging situations for the VLC channel. The transmitter tries to send a message every 0.1 s for which the agent has to select the communication link technology. It can select any combination of the DSRC radio, the VLC headlight, and the VLC taillight, yielding eight (2^3) different choices. The agent is rewarded for successful transmission, reduced by a cost based on the chosen communication links (cf. Table I). Since the interference domain of DSRC is much larger, its cost is higher than the cost of using VLC, i.e., VLC is always preferable.

This environment is implemented using our `Veins-Gym` framework described in Section III and the Veins-VLC extension [11]. The observation is encoded as the `x`- and `y`-coordinate and the orientation (as a normalized vector) of the receiver relative to the transmitter. We assume that these information are locally available to the car from its sensors. The `x` and `y` coordinates are scaled of by a factor of 10^{-3} and then clipped to the interval $[-1, 1]$. This yields values suitable for neural networks that vary on a scale similar to the normalized orientation vector.

The optimal policy maximizes reception probability, while minimizing transmission cost. The agent should eventually learn multiple things:

- When the receiver is in a defined region ahead of the transmitter and oriented towards it, using the headlight transmits the data successfully;
- communication with the taillight is never successful, because it faces away from the receiving vehicle that drives in front of the transmitting vehicle; and
- since the transmitter has exclusive access to the channel, DSRC is very reliable, albeit at a higher cost.

We implemented and compared five different policies that are used to select the communication link. Three of them are hand-written, one based on pre-sampled data, and one based on a ML model. The action selection patterns of the last four policies are illustrated in Figure 2. Note that only the RL-based policy can select between all eight communication link configurations and considers the orientation of the receiver vehicle. The other policies simply select between the DSRC and VLC headlight.

B. Hand-Written Policies

The first two policies serve as a simple baseline, and are named *DSRC only* (#1) and *VLC Head only* (#2), respectively.

Table I
KEY PARAMETERS OF THE SETUP.

VLC Radiation configuration	Veins-VLC [11]
Message Interval	0.1 s
Network structure	$\{10, 40, 75, 100\}^{2-4}$
Maximum training steps	1 000 000
Trained network instances per configuration	3
Optimizer	Adagrad
Communication Reward	1
DSRC Cost	0.5
VLC Cost	0.1

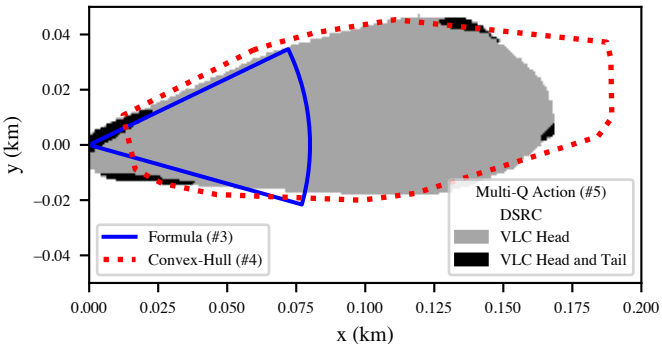

Figure 2. Patterns of selected actions for different policies for different receiver positions. The polygons indicate the region within which VLC is selected. The shaded area indicates the selected action by the Multi-Q regressor. In this case, the receiver's orientation is assumed to be the same as the transmitters.

Both only ever output the same transmission technology, regardless of the observation: The first always selects DSRC and the other always selects the VLC headlight. Both will not yield optimal rewards. But they can display the proportion of transmission attempts in which the VLC headlight is successful.

The *Formula* (#3) policy represents a simple hand-crafted heuristic. The chosen action is defined as

$$\text{link} = \begin{cases} \text{VLC}, & \text{if } \frac{-\pi}{11.5} < \alpha < \frac{\pi}{7} \wedge |d| < 80\,\text{m} \\ \text{DSRC}, & \text{else} \end{cases}, \qquad (1)$$

where α is the angle between the forward axis of the follower vehicle and d is the distance between transmitter and receiver vehicle. If the observation lies within these bounds, the VLC headlight is used, otherwise DSRC. The resulting pattern is shown as a solid blue line in Figure 2. The different angles for the left and right side represent the asymmetric radiation pattern of the headlight.

C. Data-Driven Policies

The *Convex-Hull* (#4) policy is based on 1000 pre-sampled observations of successful transmissions using the VLC headlight. A convex hull is computed from the coordinates of the pre-sampled observations. The policy then chooses the VLC headlight if a new observation falls into that convex hull and the DSRC link otherwise. An example convex hull is shown as a red dashed line in Figure 2. It does not require manual parameterization and should be able to better match the non-symmetrical transmission pattern of the headlight.

The *Multi-Q-Regressor* (#5) policy is a RL-based policy that uses eight `DNNRegressor` estimators from the Tensorflow ML framework. Each regressor uses a DNN to predict the expected reward (q-value) for one action for a given observation. The policy then selects the action with the highest predicted reward. We trained and compared different configurations of the neural networks, with a different amount of layers and nodes within each layer (cf. Table I).

In contrast to the other policies, the *Multi-Q-Regressor* policy faced additional challenges. In our very specific scenario, it has to make a more complex decision and needs to learn

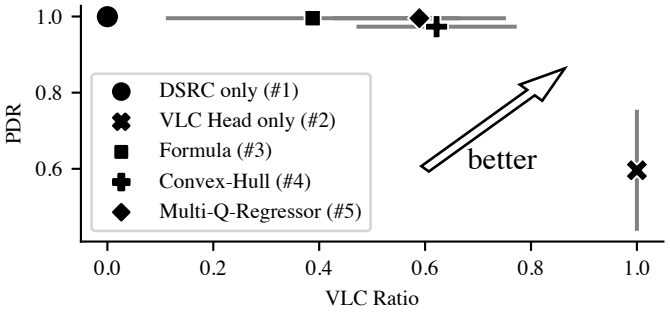

Figure 3. The performance of the strategies. The PDR is plotted against the ratio of packets sent via VLC which models the resource use. Error bars indicate standard deviation of the two metrics.

to avoid the decoy option of the VLC taillight as an unbeneficial combinations. It also has to learn that using multiple transmission links at once is generally inferior to just using one in our scenario.

V. Evaluation

We evaluated the policies described above in our `Veins-Gym` as described in Sections III and IV to determine transmission reliability and resource efficiency. Transmission reliability is expressed as Packet Delivery Ratio (PDR). As VLC is considered cheaper (cf. Section IV-A), we express transmission resource efficiency as the ratio of transmissions via VLC. Each policy is used decide the transmission link in 10 independent episodes of the environment.

We trained the *Multi-Q-Regressor* using the default *Adagrad* optimizer to control the gradient descent and learning rate adaption. We used pre-collected trajectories (records of observations, taken actions, and resulting rewards) to train the neural networks offline, though they could also be trained on-policy while exploring the environment. From the total of 36 trained networks, we picked the one with the highest average reward for comparison with the other policies. Figure 2 shows the action selected by the policy as shaded areas. This best network was configured with 4 dense hidden layers of 75 nodes each.

Figure 3 shows the performance of the five policies in the two described metrics. Both trivial policies perform as expected, *DSRC only* has near perfect reliability (99.9%) using the more limited resource of the DSRC channel. *VLC Head only* is more efficient, however it achieves only about 60% PDR within an episode on average. The *formula* policy is more efficient than *DSRC only*, using VLC for 38% of the transmissions, while achieving a similar reliability (99.6%). *Formula* makes a more conservative estimation for when VLC-based transmissions will be successful (cf. Figure 2).

The data-derived *Convex-Hull* policy on the other hand chooses to use VLC more aggressively. Accordingly, it is much more efficient (62% VLC use), but sacrifices reliability, yielding only a 97.3% PDR.

The best performing instance of the RL-based *Multi-Q-Regressor* combines both of these advantages: it employs VLC similarly often (58%), and still achieves a PDR of 99.5%. In addition to this performance, the policy has learned to avoid sub-optimal actions (i.e., using multiple communication methods) in 99.4% of the transmissions (cf. Figure 2).

VI. Conclusion

Research on VANETs can benefit for ML or RL techniques to determine optimal parameterization of protocols, or even behavior of the protocol itself. To ease the design and comparison of solutions to such problems, we designed and implemented `Veins-Gym`, an OpenAI gym for the Veins simulator. We presented an exemplary problem for selection of the optimal communication technology in a heterogeneous communication scenario, and presented different hand-crafted, learning-based, and hybrid approaches. Our learning based policy outperforms the other approaches, achieving very reliable communication while selecting the most efficient communication link. Our results indicate the potentials of using ML for ITS problems. In future work we aim to use these techniques and extend them by, e.g., multi-agent learning, to determine high performance solutions to current VANET challenges. To aid other researchers, we will release the `Veins-Gym` as Open Source.

Acknowledgments

This work has been supported in part by the German Research Foundation (DFG) under grant no. DR 639/18-1. We also acknowledge the advice by Maarten Bieshaar on RL.

References

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[2] L. Liang, H. Ye, and G. Y. Li, "Toward Intelligent Vehicular Networks: A Machine Learning Framework," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, Feb. 2019.

[3] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking Reinforcement Learning Algorithms on Real-World Robots," in *Conference CoRL 2018*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., ser. CoRL, vol. 87, Zürich, Switzerland: PMLR, Oct. 2018, pp. 561–591.

[4] E. Vinitsky, A. Kreidieh, L. L. Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Conference CoRL 2018*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., ser. CoRL, vol. 87, Zürich, Switzerland: PMLR, Oct. 2018, pp. 399–409.

[5] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *ACM MSWiM 2019*, Miami Beach, FL: ACM, Nov. 2019.

[6] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," arXiv, cs.LG 1606.01540, Jun. 2016.

[7] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *TMC*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[8] I. Zamora, N. Gonzalez Lopez, V. Mayoral Vilches, and A. Hernandez Cordero, "Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo," arXiv, cs.RO 1608.05742, Feb. 2017.

[9] C. Choe, J. Choi, J. Ahn, D. Park, and S. Ahn, "Multiple Channel Access using Deep Reinforcement Learning for Congested Vehicular Networks," in *IEEE VTC 2020-Spring*, Virtual Conference: IEEE, May 2020.

[10] A. Pressas, Z. Sheng, F. H. Ali, and D. Tian, "A Q-Learning Approach With Collective Contention Estimation for Bandwidth-Efficient and Fair Access Control in IEEE 802.11p Vehicular Networks," *TVT*, vol. 68, no. 9, pp. 9136–9150, Sep. 2019.

[11] A. Memedi, C. Tebruegge, J. Jahneke, and F. Dressler, "Impact of Vehicle Type and Headlight Characteristics on Vehicular VLC Performance," in *IEEE VNC 2018*, Taipei, Taiwan: IEEE, Dec. 2018.