Deep Reinforcement Learning for Dynamic Urban Transportation Problems

Laura Schultz and Vadim Sokolov
Systems Engineering and Operations Research
George Mason University. Fairfax, VA
Email: {lschult2, vsokolov}@gmu.edu

***Abstract*—Many transportation system analysis tasks are formulated as an optimization problem - such as optimal control problems in intelligent transportation systems and long term urban planning. The models often used to represent the dynamics of a transportation system involve large data sets with complex input-output interactions and are difficult to use in the context of optimization. We explore the use of deep learning and deep reinforcement learning for such optimization problems in transportation. Use of deep learning meta-models can produce a lower dimensional representation of those relations and allow to implement optimization and reinforcement learning algorithms in an efficient manner. In particular, we develop deep learning models for calibrating transportation simulators and reinforcement learning to solve the problem of optimal scheduling of travelers on the network.**

I. Introduction

Many modern transportation system analysis problems, such as fleet management [1], intelligent system operations [2] and long-term urban planning [3], lead to high-dimensional and highly nonlinear optimization problems. Analytical formulations using mathematical programming [4], [5] or conservation laws [6], [7] are typically assumed but rely on high level abstractions, such as origin-destination matrices for demand. Alternatively, complex simulation [8], [9], [10] models, which model individual travelers through Agent-Based Modeling (ABM), provide a flexible approach to represent traffic and demand patterns in large scale multi-modal transportation systems. However, the computational costs often times prove prohibitive and meta-model-based approaches have been proposed [11], [12].

In this paper, we propose an alternative approach to solving these optimization problems for large scale transportation systems. Our approach relies on deep learning approximators, a Latent Variable Model (LVM) technique which is capable of extracting the underlying low-dimensional pattern in high-dimensional input-output relationship. Deep learners have proven highly effective in combination with Reinforcement and Active Learning [13] to recognize these latent patterns for exploitation. Our approach builds on the work of simulation-based optimization [11], [12], deep learning [14], [15], as well as reinforcement learning [16], [17] techniques recently proposed for transportation applications. The two main contribution of this paper are

1) Development of innovative deep learning architecture for reducing dimensionality of search space and modeling relations between transportation simulator inputs (travel behavior parameters, traffic network characteristics) and outputs (mobility patterns, traffic congestion)
2) Development of reinforcement learning techniques that rely on deep learning approximators to solve optimization problems for dynamic transportation systems

We demonstrate our methodologies using two applications. First, we solve the problem of calibrating a complex, stochastic transportation simulator to accurately match recorded field data in order to make the representation useful for both short term operational decisions and long term urban planning. Leveraging previously proposed methods, which treat the problem as an optimization issue [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], our approach makes no assumptions about the form of the simulator and types of inputs and outputs used. Further, we show that deep learning models are more sample efficient when compared to unadulterated Bayesian techniques or traditional dimension-reduction methods. We build on our calibration framework [12] by further exploring the dimensionality reduction utilized for more efficient input parameter space exploration. More specifically, we introduce the formulation and analysis of a combinatorial Neural Network method and compare it with previous work that used Active Subspace methods.

The second application builds upon recent advances in deep learning approaches to Reinforcement Learning (RL). Demonstrating impressive results in many applications [28], [29] through neural network approximations of state-action functions, RL mimics the way humans learn new tasks and behavioral policies via trial and error. Most research on RL has been concentrated to the field of machine learning and classical Artificial Intelligence (AI) problems, such as robotics, language translation and supply chain management problems [30]; however, some classical transportation control problems have been previously solved using RL [31], [32], [33], [34], [31], [35], [36], [37], [38], [39]. Furthermore, the deep RL has was recently applied and proven successful for traffic flow control [40], [17], [16], [41], [42].

The remainder of this paper is organized as follows: Section II briefly documents the highlights of neural network architectures; Section II-C describes the new deep learning architecture that finds low dimensional patterns in simulator's inputs-output relations and we apply our deep learner to the problem of model calibration. Section III describes

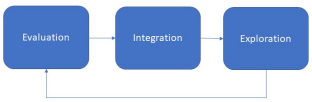


Fig. 1. Three Stages of Calibration Framework

the additional application of deep reinforcement learning to transportation system optimization. Finally Section IV offers avenues for further research.

II. Deep Learning for Calibration

The primary goal of a calibration procedure is to find a set of static input parameters that lead to simulation outputs that match observed data, such as traffic counts, point-to-point travel times, or transit ridership, as close as possible. A simulator that is capable of reproducing observed flows can then be used confidently to develop forecasts on network performance for different infrastructure changes or policies.

Recently, we developed a meta-model framework, which executes Bayesian optimization algorithms in distributed computing environments, to be used for these transportation simulation-based problems [12]. Decomposed into three re-iterative stages, as shown in Figure 1, the framework takes a generalizable approach by using black-box optimization methodologies when the simulator is highly complex but the resources for evaluation are limited; Specifically, a Gaussian Process (GP) approach constructs a probability distribution over all potential linear and nonlinear functions representing the discrepancy relationship between the simulator and field data and seeks a valued estimation for the simulator's static input parameters with minimal uncertainty and resources.

Unfortunately, Bayesian models do not scale favorably with dimensionality. The contribution of a single sample to the understanding of evaluation space decreases as the space between samples becomes larger and dimensions grow. In practice, however, high dimensional data possesses a natural structure within it that can be successfully expressed in lower dimensions. Known as dimension reduction, the effective number of parameters in the model reduces and enables successful analysis from smaller data sets.

The previous work utilized a reduction technique known as Active Subspaces to address this concern with limited success. In this section, we develop a new, combinatorial deep learning architecture that can be used to learn the low-dimensional structure instead.

A. Multi-Layer Perceptron

Let y denote a multi-dimensional output and $x = (x_1, \ldots, x_p) \in R^p$ a (high dimensional) set of inputs. We wish to recover the multivariate function (map), denoted by $y = f(x)$, using training data of input-output pairs $(y_i, x_i)_{i=1}^N$, that generalizes well for out-of-sample data. DL is a pattern matching model that uses composition of univariate functions to approximate the input-output relations. By composing L layers, a deep learning predictor, known as a Multi-layer Perceptron network, becomes

$$\begin{aligned}\hat{y} =& F_{W,b}(x) = (f^0_{w_0,b_0} \circ \ldots \circ^L_{w_L,b_L})(x)\\ f^i_{w_i,b_i} =& f_i(w_i x_i + b_i) \quad \forall \quad i \in [0, L].\end{aligned}$$

Here f_i is a univariate activation function. Wights $w_l \in R^{p_l \times p_l'}$ and offsets $b_l \in R$ are selected by applying stochastic gradient descent (SGD) to solve a regularized least squares optimization problem given by

$$\underset{W,b}{\text{minimize}} \sum_{i=1}^{N} ||y_i - F_{W,b}(x_i)||_2^2 + \phi(W, b).$$

Here ϕ is a regularization penalty on the network parameters (weights and offsets).

B. Auto-Encoder

An auto-encoder is a deep learning routine which trains the architecture to approximate x by itself (i.e., $x = y$) via a bottleneck structure. This means we select a model $F_{W,b}(x)$ which aims to concentrate the information required to recreate x. Put differently, an auto-encoder creates a more cost effective representation of x. For example, under an L_2-loss function, we wish to solve

$$\underset{W,b}{\text{minimize}} \, ||F_{W,b}(x) - x||_2^2$$

subject to a regularization penalty on the weights and offsets. In an auto-encoder, for a training data set $\{x_1, \ldots, x_n\}$, we set the target values as $y_i = x_i$. A static auto-encoder with two linear layers, akin to a traditional factor model, can be written via a deep learner as

$$\begin{aligned}a^{(1)} =& x\\ a^{(2)} =& f_1(w^{(2)} a^{(1)} + b^{(2)})\\ a^{(3)} =& F_{W,b}(x) = f_2(w^{(3)} a^{(2)} + b^{(3)}),\end{aligned}$$

where $a^{(i)}$ are activation vectors. The goal is to find the weights and biases so that size of $w^{(2)}$ is much smaller than size of $w^{(3)}$.

C. Deep Learning Architecture

Within the calibration framework, two objectives must be realized by the neural network:

1) A reduced dimension subspace which captures the relationship between the simulator inputs and outputs must be found and bounded in order for adequate exploration of the state-space to determine the next useful evaluation point
2) Given that the recommended evaluation points are expressed in the reduced dimension sample space, the network must be able to recover a projected sample back to the original space to allow for simulator evaluation

Given the deterministic nature of our DL model [15], we use the neural network for only dimensionality reduction in our GP calibration framework.

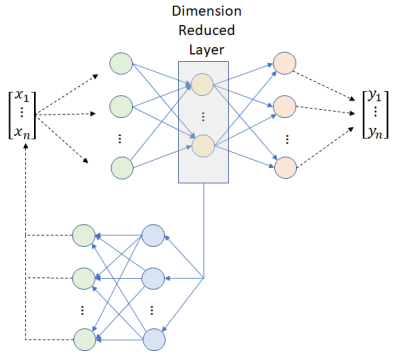


Fig. 2. Graphical Representation of the Combinatorial Neural Network for Calibration

To address these objectives, we combine an MLP architecture to capture a low-dimensional representation of the input-output with an autoencoder architecture to decode the new subspace back to the original. We will then be able to run the calibration framework's optimization algorithms inside the low dimensional representation of the input parameter space to address the curse of dimensionality issue and convert the recommended sample into a usable form for simulator evaluation. The Autoencoder and MLP share the same initial layers up to the reduced dimension layer, as shown in Figure 2.

The activation function used is the $\tanh$, which provides a bounded range of $(-1, 1)$ and is a close approximation to the sine function.

We ran the simulator several times to generate an initial sample set for use as training data to explore the relationship between the inputs and outputs; a hypercube sampling across the original dimensions subspace is utilized. Additionally, to quantify the discrepancies during training of the neural network, the following loss functions are used:

1) The MLP portion of the architecture uses the mean squared error function L

$$L\left[y, \phi(\theta)\right] = \frac{1}{N}\sum_{i=1}^{N}(y - \phi(\theta_i))^2 \tag{1}$$

where $\phi(\theta)$ represents the predicted values produced by the neural network for the simulator's output y given the input set θ

2) The Autoencoder portion of the architecture uses the mean squared error function L and a quadratic penalty cost P for producing predicted values outside of the original subspace bounds since the simulator cannot evaluate a point that does not lie in the plausible domain

$$D\left[\phi(\theta)\right] = \max[0, \phi(\theta_i) - x_u]^2 + \max[0, x_l - \phi(\theta_i)]^2 \tag{2}$$

where $\phi(\theta)$ represents the predicted values produced by the neural network for the simulator's input x given the input set θ, x_u represents the input set's upper bound, and x_l represents the input set's lower bound

D. Empirical Results

We use Sioux-Falls [43], a transportation model consisting of 24 intersections with 76 directional roads, or arcs, for our empirical results. The network structure and input data provided by [44] have been adjusted from the original dataset to approximate hourly demand flows in the form of Origin-Destination (O-D) pairs, the simulation's input set.

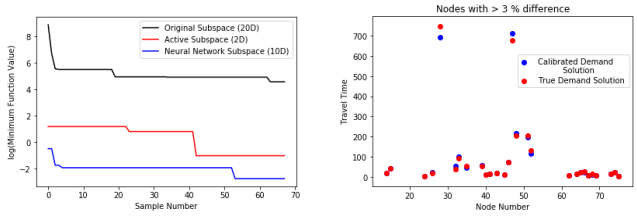
The input data is provided to a simulator package which implements the iterative Frank-Wolfe method to determine the traffic equilibrium flows and outputs average travel times across each arc. Due to limited computing availability, only the first twenty O-D pairs are treated as unknown input variables between 0 and 7000, which need to be calibrated, while the other O-D pairs are assumed to be known and fixed. Random noise is added to the simulator to emulate the observational and variational errors expected in real-world applications. The calibration framework's objective function is to minimize the mean discrepancy between the simulated travel times resulting from the calibrated O-D pairs and the 'true' times resulting from the full set of true O-D pair values.

The neural network begins with an input layer, f_0, of the 20 O-D input pairs, $x_{1:20}$, we are attempting to calibrate. Using a Tanh activation function, three layers, f_1, f_2, f_3, with 10 nodes each produce the internal layers of the deep network connecting the input and the first output layer, $y_{1:76}$, which contains the average traffic times across each of the 76 arcs. Three additional layers, using a Tanh activation function as well, with 10 nodes each, f_4, f_5, f_6, connect the second middle layer, f_2, to a secondary output layer consisting of the reconstructed x input values. See Figure 2 for a visualization of the architecture used.

Overall, the performance of the calibration using a deep neural network proved significant, see Figure 3(a). A calibrated solution set was produced which resulted in outputs, on average, within 3% of the experiment's true output. With a standard deviation of 5%, Figure 3(b) provides a visualization for those links which possessed greater than average variation from the true demand's output. Given the same computational budget, Bayesian optimization that uses low dimensional representation from the deep learner leads to 25% more accurate match between measured and simulated data when compared to active subspaces.

III. Deep Reinforcement Learning

Consider the desire for a calibrated simulator not to be used for the evaluation of interested scenarios but as a tool for designing a policy $\pi : s \rightarrow a$ which dictates an optimal action a for the current state of the system s. A simulator is an interactive system of players (travelers, system operators, etc.), known as agents, and their environment. In such a system, the agent interacts with an environment in discrete time steps. At each timestep, t, the agent has a set of actions, A which can be executed. Given the action, the environment changes from its original state, s, to a new, influenced state, s'. Reinforcement Learning (RL) allows to find a sequence of actions to achieve



(a) Objective Function (b) Calibrated vs True

Fig. 3. Results of demand matrix calibration using Bayesian optimization. (a) Comparison of the Three Methods in terms of Objective Evaluations applied to original parameter space (black line), reduced dimensionality parameter space of Active Subspaces (red line), and reduced dimensionality parameter space of Neural Networks(blue line). (b) Comparison of Calibrated and True Travel Time Outputs with Above Average Differences.

the desirable state. The desirability of a state is modeled by the reward function.

This approach is quite conducive to transportation. For example, if a commuter chooses to leave the house after rush hour has ended, he will eventually be rewarded at the end of his commute with a shorter travel. Although not immediately realized, the reward is no less desired and will, in the future, encourage the agent to apply the same actions when possible.

One approach to RL is via Q-learning [45]. Q-learning, represents the 'quality' of a certain action within the environment's current state via a Q-function, hich represents the maximum, discounted reward that can be obtained in the future if action a is performed in state s and all subsequent actions are continued following the optimal policy π from that state on:

$$Q_\pi(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi],$$

where $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$ is the discounted return and $\tau \in [0,1]$ is the factor used to enumerate the importance of immediate and future rewards.

In other words, it is the greatest reward we can expect given we follow the best set of action sequences after performing action a in state s. Subsequently, the optimal policy requires choosing the optimal, or maximum, value for each state:

$$Q^*(s,a) = \max_\pi Q_\pi(s,a)$$

Unfortunately, the Q-functions for these transportation simulators continue to possess high-dimensionality concerns similarly noted in our previous calibration work. However, recent advancements have allowed for the successful integration of reinforcement learning's Q-function with deep neural networks [46]. Known as a Deep Q Network (DQN), these neural networks have the potential to provide a diminished feature set for highly structured, highly-dimensional data without hindering the power of the reinforcement learning.

For development and training of such a network, a neural network architecture best-fitting the problem is constructed with the following loss function [47]

$$L_i(\theta_i) = E_{s,a,r,s'}\left[(y_i^{DQN} - Q(s,a;\theta_i))^2\right],$$

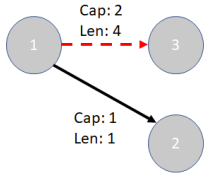


Fig. 4. Graphical Representation of the Example Transportation System

where θ are the parameters, $Q(\cdot)$ is the Q-function for state s and action a and

$$y_i^{DQN} = r + \gamma \max_{a'} Q(s', a'; \theta^-),$$

where θ^- represents parameters of a fixed and separate target network.

Furthermore, to increase the data efficiency and reduce the correlation among samples during training, DQNs leverage a buffer system known as *experience replay*. Each transition and answer set, (s_t, a_t, r_t, s_{t+1}), is stored offline and, throughout the training process, random small batches from the replay memory are used instead of the most recent transition. For the purpose of this paper, a MLP network is utilized as the neural architecture.

A. Empirical Results

For demonstration and analysis, a small transportation network, depicted in Figure 4, consisting of 3 nodes and 2 routes, or arcs, is used.

The small network has varying demand originating from node 1 to node 2 for 24 time periods. Using RL, we find the best policy to handle this demand with the lowest overall system travel time given that any single period has two allowable actions:

1) $0-2$ units of demand from node 1 to node 2 can be delayed up to one hour
2) $0-2$ units of demand from node 1 to node 2 can be re-routed to node 3 as an alternative destination at a further distance

In essence, we solve the optimal traffic assignment problem. Our state contains the following information: (i) the amount of original demand from node 1 to node 2 that is to be executed at time t, $D_{t,1,2}$; (ii) the amount of demand moved to time t for execution from time $t-1$, $M_{t,1,2}$; (iii) the amount of demand left to be met between time $t+1$ and $t=24$, divided by the amount of time left $24-(t+1)$.[1] The action set includes the option to move 0,1,or 2 units of demand from the current period t to the subsequent period $t+1$ or move 0,1,or 2 units of demand from the arc between note 1 and 2 to the arc between node 1 and 3 , $A_t = [A_{t,t+1,1,2}, A_{t,1,3}]$. The reward is calculated using the same simulator package from the Section II-D, which implements the iterative Frank-Wolfe method to determine the traffic equilibrium flows and outputs

[1]Q-learning requires a Markov Process assumption. The third state value prevents the solution from attempting to delay demand as long as possible while meeting the ergodic requirement of Markov Processes

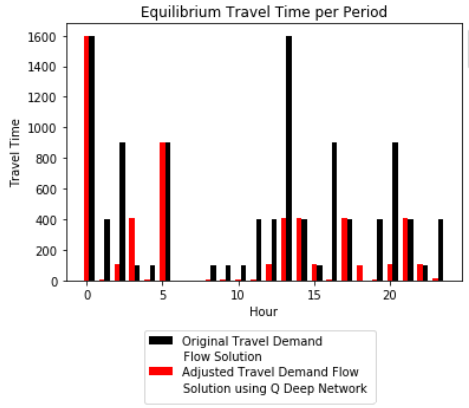


Fig. 5. Comparison of System Travel Time per Period

total system travel time for the period. Since Q-learning seeks the maximum reward, we took the negative total system travel time over the 24 periods as the reward.

Four inputs are given to the neural network to describe the state: original demand, moved demand, demand left, and the current time t. These inputs are then fed into two hidden layers with a $\tanh$ activation function consisting of ten nodes each and outputs to the nine combinatorial outputs possible for the two allowable actions. The amount the discount factor,γ, was set to 0.9 and the action selection is done according to an ϵ-greedy policy, which takes the highest value according to the neural network or picks randomly at a probability that decays exponentially, in order to initially encourage exploration.

After running the network on 100 of 24-long episodes, a randomly generated set of demand was produced and run through the resulting neural network. A 51% improvement in the system travel time was achieved. Table I illustrates the adjustments decided by the Q network and Figure 5 compares the travel times by period between the original and adjusted demands.

TABLE I
ADJUSTMENTS TO DEMANDS PER PERIOD USING DQN

Hour	Original Demand of $\mathbf{Arc_{1,2}}$	DQN Adjusted Demand of $\mathbf{Arc_{1,2}}$	DQN Adjusted Demand of $\mathbf{Arc_{1,3}}$
1	4	4	0
2	2	0	1
3	3	1	1
4	1	2	1
5	1	0	1
6	3	3	0
7	0	0	0
8	0	0	0
9	1	0	1
10	1	0	1
11	1	0	1
12	2	0	1
13	2	1	1
14	4	2	1
15	2	2	1
16	1	1	1
17	3	0	1
18	2	2	1
19	0	1	0
20	2	0	1
21	3	1	1
22	2	2	1
23	1	1	1
24	2	0	2

IV. DISCUSSION

Deep learning provides a general framework for modeling complex relations in transportation systems. As such, deep learning frameworks are well-suited to many optimization problems in transportation. This paper presents an innovative deep learning architecture for applying reinforcement learning and calibrating a transportation model. We have demonstrated, deep learning is a viable option compared to other metamodel based approaches. Our calibration and reinforcement learning examples demonstrate how to develop and apply deep learning models in transportation modeling.

At the same time, there are significant challenges associated with using deep learning for optimization problems. Most notably, the issue of performance of deep reinforcement learning [48]. Though theoretical bounds on performance of different RL algorithms do exist, the research done over the past few decades showed that worst case analysis is not the right framework for studying artificial intelligence: every model that is interesting enough to use in practice leads to computationally hard problems [49]. Similarly, while there are many important theoretical results that show very slow convergence of many RL algorithm, it was shown to work well empirically on specific classes of problems. The convergence analysis developed for RL techniques is usually asymptotic and worst case. Asymptotic optimality was shown by [45] who shows that Q-learning, which is an iterative scheme to learn optimal policies, does converge to optimal solution Q^* asymptotically. Littman et.el. [50] showed that a general reinforcement learning models based on exploration model does converge to an optimal solution. It is not uncommon for convergence rates in practice to be much better than predicted by worst case scenario analysis. Some of the recent work suggests that using recurrent architectures for Value Iteration Networks (VIN) can achieve good empirical performance compared to fully connected architectures [51]. Adaptive approaches that rely on meta-learning were shown to improve performance of reinforcement learning algorithms [52].

Another issue that requires further research is the bias-variance trade-off in he context of deep reinforcement learning. Traditional regularization techniques that add stochasticity to RL functions do not prevent from over-fitting [53].

In the meantime, deep learning and deep reinforcement learning are likely to exert greater and greater influence in the practice of transportation.

REFERENCES

[1] R. Nair and E. Miller-Hooks, "Fleet management for vehicle sharing operations," *Transportation Science*, vol. 45, no. 4, pp. 524–540, 2011.

[2] K. Lam, W. Krichene, and A. Bayen, "On learning how players learn: estimation of learning dynamics in the routing game," in *Cyber-Physical Systems (ICCPS), 2016 ACM/IEEE 7th International Conference on*. IEEE, 2016, pp. 1–10.

[3] H. Spiess and M. Florian, "Optimal strategies: a new assignment model for transit networks," *Transportation Research Part B: Methodological*, vol. 23, no. 2, pp. 83–102, 1989.

[4] J. Larson, T. Munson, and V. Sokolov, "Coordinated platoon routing in a metropolitan network," in *2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*. SIAM, 2016, pp. 73–82.

[5] V. Sokolov, J. Larson, T. Munson, J. Auld, and D. Karbowski, "Maximization of platoon formation through centralized routing and departure time coordination," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2667, pp. 10–16, 2017.

[6] C. G. Claudel and A. M. Bayen, "Lax–hopf based incorporation of internal boundary conditions into hamilton–jacobi equation. part i: Theory," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1142–1157, 2010.

[7] N. Polson, V. Sokolov *et al.*, "Bayesian analysis of traffic flow on interstate i-55: The lwr model," *The Annals of Applied Statistics*, vol. 9, no. 4, pp. 1864–1888, 2015.

[8] K. Nagel and G. Flätterd, "Agent-based traffic assignment: Going from trips to behavioural travelers," in *Travel Behaviour Research in an Evolving WorldSelected papers from the 12th international conference on travel behaviour research*. International Association for Travel Behaviour Research, 2012, pp. 261–294.

[9] V. Sokolov, J. Auld, and M. Hope, "A flexible framework for developing integrated models of transportation systems using an agent-based approach," *Procedia Computer Science*, vol. 10, pp. 854–859, 2012.

[10] J. Auld, M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang, "Polaris: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 101–116, 2016.

[11] L. Chong and C. Osorio, "A simulation-based optimization algorithm for dynamic large-scale urban transportation problems," *Transportation Science*, 2017.

[12] L. Schultz and V. Sokolov, "Bayesian optimization for transportation simulators," *Procedia Computer Science*, vol. 130, pp. 973–978, 2018.

[13] 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, p. 529, 2015.

[14] M. F. Dixon, N. G. Polson, and V. O. Sokolov, "Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading," *arXiv preprint arXiv:1705.09851*, 2017.

[15] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.

[16] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, 2017.

[17] C. Wu, K. Parvate, N. Kheterpal, L. Dickstein, A. Mehta, E. Vinitsky, and A. M. Bayen, "Framework for control and deep reinforcement learning in traffic," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–8.

[18] R. Cheu, X. Jin, K. Ng, Y. Ng, and D. Srinivasan, "Calibration of FRESIM for Singapore expressway using genetic algorithm," *Journal of Transportation Engineering*, vol. 124, no. 6, pp. 526–535, Nov. 1998.

[19] T. Ma and B. Abdulhai, "Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters," *Intelligent Transportation Systems and Vehicle-highway Automation 2002: Highway Operations, Capacity, and Traffic Control*, no. 1800, pp. 6–15, 2002.

[20] L. Lu, Y. Xu, C. Antoniou, and M. Ben-Akiva, "An enhanced SPSA algorithm for the calibration of Dynamic Traffic Assignment models," *Transportation Research Part C: Emerging Technologies*, vol. 51, pp. 149–166, Feb. 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0968090X14003295

[21] E. Cipriani, M. Florian, M. Mahut, and M. Nigro, "A gradient approximation approach for adjusting temporal origindestination matrices," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 2, pp. 270–282, Apr. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0968090X10000975

[22] J. B. Lee and K. Ozbay, "New calibration methodology for microscopic traffic simulation using enhanced simultaneous perturbation stochastic approximation approach," *Transportation Research Record*, no. 2124, pp. 233–240, 2009.

[23] D. K. Hale, C. Antoniou, M. Brackstone, D. Michalaka, A. T. Moreno, and K. Parikh, "Optimization-based assisted calibration of traffic simulation models," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 100–115, Jun. 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0968090X15000261

[24] M. L. Hazelton, "Statistical inference for time varying origindestination matrices," *Transportation Research Part B: Methodological*, vol. 42, no. 6, pp. 542–552, Jul. 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0191261507001348

[25] G. Flätterd, "A general methodology and a free software for the calibration of DTA models," in *The Third International Symposium on Dynamic Traffic Assignment*, 2010.

[26] G. Flätterd, M. Bierlaire, and K. Nagel, "Bayesian demand calibration for dynamic traffic simulations," *Transportation Science*, vol. 45, no. 4, pp. 541–561, 2011.

[27] T. Djukic, G. Flätterd, H. Van Lint, and S. Hoogendoorn, "Efficient real time OD matrix estimation based on Principal Component Analysis," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 115–121.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2017.

[30] I. Giannoccaro and P. Pontrandolfo, "Inventory management in supply chains: a reinforcement learning approach," *International Journal of Production Economics*, vol. 78, no. 2, pp. 153–161, 2002.

[31] B. Abdulhai and L. Kattan, "Reinforcement learning: Introduction to theory and potential for transport applications," *Canadian Journal of Civil Engineering*, vol. 30, no. 6, pp. 981–991, 2003.

[32] T. Arentze and H. Timmermans, *Albatross: a learning based transportation oriented simulation system*. Eirass Eindhoven, 2000.

[33] E. Bingham, "Reinforcement learning in neurofuzzy traffic signal control," *European Journal of Operational Research*, vol. 131, no. 2, pp. 232–241, 2001.

[34] A. L. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, pp. 342–375, 2009.

[35] 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.

[36] K. Ling and A. S. Shalaby, "A reinforcement learning approach to streetcar bunching control," *Journal of Intelligent Transportation Systems*, vol. 9, no. 2, pp. 59–68, 2005.

[37] R. Cunningham, A. Garg, V. Cahill *et al.*, "A collaborative reinforcement learning approach to urban traffic control optimization," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, vol. 2. IEEE, 2008, pp. 560–566.

[38] Z. Adam, M. Abbas, and P. Li, "Evaluating Green-Extension Policies with Reinforcement Learning and Markovian Traffic State Estimation," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2128, pp. 217–225, Dec. 2009. [Online]. Available: http://trrjournalonline.trb.org/doi/abs/10.3141/2128-22

[39] L. Chong, M. Abbas, B. Higgs, A. Medina, and C. Y. D. Yang, "A revised reinforcement learning algorithm to model complicated vehicle continuous actions in traffic," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2011, pp. 1791–1796.

[40] Z. Adam, M. Abbas, and P. Li, "Evaluating green-extension policies with reinforcement learning and markovian traffic state estimation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2128, pp. 217–225, 2009.

[41] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *arXiv preprint arXiv:1701.08832*, 2017.

[42] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[43] L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla, "An efficient approach to solving the road network equilibrium traffic assignment problem," *Transportation research*, vol. 9, no. 5, pp. 309–318, 1975.

[44] B. Stabler, "TransportationNetworks: Transportation Networks for Research," Sep. 2017, original-date: 2016-03-12T22:38:10Z. [Online]. Available: https://github.com/bstabler/TransportationNetworks

[45] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992. [Online]. Available: https://link.springer.com/article/10.1007/BF00992698

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://www.nature.com/articles/nature14236

[47] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," *arXiv:1511.06581 [cs]*, Nov. 2015, arXiv: 1511.06581. [Online]. Available: http://arxiv.org/abs/1511.06581

[48] C. Wu, A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. Kakade, I. Mordatch, and P. Abbeel, "Variance reduction for policy gradient with action-dependent factorized baselines," *arXiv preprint arXiv:1803.07246*, 2018.

[49] A. Bhaskara, M. Charikar, A. Moitra, and A. Vijayaraghavan, "Smoothed analysis of tensor decompositions," *CoRR*, vol. abs/1311.3651, 2013. [Online]. Available: http://arxiv.org/abs/1311.3651

[50] M. L. Littman and C. Szepesvari, "A Generalized Reinforcement-Learning Model: Convergence and Applications," Brown University, Providence, RI, USA, Tech. Rep., 1996.

[51] L. Lee, E. Parisotto, D. S. Chaplot, and R. Salakhutdinov, "Lstm iteration networks: An exploration of differentiable path finding," 2018.

[52] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," *arXiv preprint arXiv:1710.03641*, 2017.

[53] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A Study on Overfitting in Deep Reinforcement Learning," *ArXiv e-prints*, Apr. 2018.