Deep Reinforcement Learning for Dynamic Urban Transportation Problems

***Abstract*—We explore the use of deep learning and deep reinforcement learning for optimization problems in transportation. 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. Often transportation models used to represent dynamics of a transportation system involve large data sets with complex input-output interactions and are difficult to use in the context of optimization. Use of deep learning metamodels 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 for reinforcement learning to solve the problem of optimal scheduling of travelers on the network.**

I. Introduction

Many modern transportation system analysis problems lead to high-dimensional and highly nonlinear optimization problems. Examples include fleet management [1], intelligent system operations [2] and long-term urban planning [3]. Transportation system models used in those optimization problems typically assume analytical formulations using mathematical programming [4], [5] or conservation laws [6], [7] and rely on high level abstractions, such as origin-destination matrices for demand. An alternative approach is to model transportation system using a complex simulator [8], [9], [10], which model individual travelers and provide flexible approach to represent traffic and demand patterns in large scale multi-modal transportation systems. However, solving optimization problems and dynamic-control problems that rely on simulator models of the system is prohibitive due to computational costs. Recently matamodel based approach was proposed to solve simulation-based transportation optimization problems [11], [12].

In this paper, we propose an alternative approach to solve optimization problems for large scale transportation systems. Our approach relies on low complexity metamodels deduced from complex simulators and reinforcement learning to solve optimization problems. We use deep learning approximators for the low complexity metamodels. A deep learning is a Latent Variable Model (LVM) that is capable of extracting the underlying low-dimensional pattern in a high-dimensional input-output relations. Deep learners have proven highly effective in combination with Reinforcement and Active Learning [13] to recognize such patterns for exploitation. Our approach builds on the work on simulation-based optimization [12], [11], 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 behavour 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 simulators which need to be systematically adjusted to match field data. The problem of calibrating a simulator is the key for making it useful for both short term operational decisions and long term urban planning. We improve on the previously proposed numerous approaches for the calibration of simulation-based traffic flow models have been produced by treating the problem as an optimization issue [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. Our approach makes no assumption 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 Bayesian techniques or more traditional filtering techniques. 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) that have demonstrated impressive results in game playing [28] through the application of neural networks for approximating state-action functions. Reinforcement Leaning mimics the way humans learn new tasks and behavioral policies via trial and error and has proven successful [29] in many applications. While most of the research on RL is done in the field of machine learning and applied to classical AI problems, such as robotics, language translation and supply chain management problems [30], some classical transportation control problems have been previously solved using RL. [31], [32], [33], [34], [31], [35], [36], [37], [38], [39]. Furthermore, there were recent attempts that successfully demonstrated applications of deep RL to traffic flow control [40], [16], [17], [41], [42].

The remainder of this paper is organized as follows: Section II briefly documents the highlights of neural network architectures; Section III describes the new deep learning architecture that finds low dimensional patterns in simula-

tor's inputs-output relations and we apply our deep learner to the problem of model calibration. Section IV describes the additional application of deep reinforcement learning to transportation system optimization. Finally Section V offers avenues for further research.

II. Deep Learning

Let y denote a (low dimensional) output and $x = (x_1, \ldots, x_p) \in \Re^p$ 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 $(x_i, f(x_i))_{i=1}^N$, that generalizes well for out-of-sample data. Deep learning uses compositions of functions, rather than traditional additive ones. By composing L layers, a deep learning predictor becomes

$$\begin{aligned} \hat{y} =& F_{w,b}(x) = (f_{w_0,b_0} \circ \ldots f_{w_L,b_L})(x_1, \ldots, x_p) \\ f^l_{w_l,b_l} =& f_l(w_l x_l + b_l). \end{aligned}$$

Here f_l is a univariate activation function. The set $(w, b) = (w_1, \ldots, w_L, b_1, \ldots, b_L)$ is the set of weights and offsets which are learned from training data. Here $w_l \in R^{p_l \times p_l'}$ and dimensionality p_l is of the architecture specifications.

Training the parameters (W, b) and selecting an architecture is achieved by regularized least squares. Stochastic gradient descent (SGD) and its variants are used to find the solution to the optimization problem [14].

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

where $(Y_i, x_i)_{i=1}^N$ is training data of input-output pairs, and $\phi(f_{\text{DL}})$ is a regularisation penalty on the network parameters (weights and offsets).

In this paper we develop a new deep learning architecture for simultaneously learning low dimensional representation of simulator's input parameter space as well as the relation between simulator's inputs and outputs. Our architecture relies on multi-layer perceptron and auto-encoding layers. Multilayer Perceptron Network (MLP) – a neural network which takes a set of inputs, $x = (x_1, x_2, ...x_i)^T$, and feeds them through one or more sets of intermediate layers to compute one or more output values, $Y = (y_1, y_2, ...y_n)^T$. Although a single architecture is commonly implemented in practice, some success has been found through comparative or combinatorial means[43], [44].

A. 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 L2-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 as 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 wind the weights and biases so that size of $w^{(2)}$ us much smaller than size of $w^{(3)}$

III. Deep Learning for Calibration

In this section we develop a new deep learning architecture that can be used to learn low-dimensional structure in the simulator input-output relations. Then we use the low dimensional representations to solve an optimization problem. Our optimization problem is the problem of calibration of a transportation simulator. As the simulators become more detailed, the high dimensionality has now become a pressing concern. In practice, high dimensional data possesses a natural structure within it, that can be expressed in low dimensions. Known as Dimension Reduction, the effective number of parameters in the model reduces and enables analysis from smaller data sets.

Recently, we developed a low-dimensional metamodel for to be used for transportation simulation-based problems [12]. We calculated active subspaces to capture low-dimensional structures in the simulator. We used Gaussian process to represent the input-output relations. There are several key concerns while developing an efficient simulation-based algorithms for transportation applications

1) Algorithms must be sample efficient and parallelizable. Each simulation run is computationally expensive and can take up to a few days. This computational constraint could potentially limit the scale and scope of calibration investigations and result in large areas of sample space unexplored and sub-optimal decisions. As High-Performance Computing (HPC) resources have become increasingly available in most research environments, new modes of computational processing and experimentation have become possible – parallel tasking capabilities allow multiple simulated runs to be performed simultaneously and HPC programs aid in coordinating worker units to run codes across multiple processors to maximize the available resources and time management. By leveraging these advances and running a queue of pending input sets concurrently through the simulator, a larger set of unknown inputs can be evaluated in an acceptable time frame.
   The variational landscape for a simulation model will not be uniform throughout the state-space. Although active conservation of restricted resource allocations can be mitigated by HPC, additional care should be taken

to determine when exploration or exploitation should be encouraged given the data collected and redundant sampling avoided.
A machine learning technique known as *active learning* is leveraged to provide such a scheme. A utility function (a.k.a acquisition function) is built to balance the exploration of unknown portions of the input sample space with the exploitation of all information gathered by the previous evaluated data, resulting in a prioritized ordering reflecting the motivation and objectives behind the calibration effort. The expectation of the utility function is taken over the Bayesian posterior distribution and maximized to provide a predetermined number of recommendations.

2) Transportation modelers have many ways to model complex interactions represented within the transportation simulator. Calibration methodologies that account for internal structure of a simulator could be more efficient. On the other hand, they are hardly generalizable to other types of simulation models. By treating the relationship between the inputs and outputs of the simulator in question as an unknown, stochastic function, black-box optimization methodologies can be leveraged. Specifically, our previously developed Gaussian process framework took the Bayesian approach to construct a probability distribution over all potential linear and nonlinear functions matching the simulator and leveraging evidential data to determine the most likely match. Once this distribution is sufficiently mapped, a valued estimation for the sought parameters can be made with minimal uncertainty.

A. 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 bounded in order for adequate exploration
2) Given the reduced dimension sample determined by the framework, a method to convert the reduction to the original dimension subspace must exist to allow for simulator evaluations

To address these objectives, we use MLP architecture to capture the relations between inputs and outputs. We use an Autoencoder architecture to capture the low dimensional structure in the input parameters. We will run optimization algorithms inside the low dimensional representation of the input parameter space to address the curse of dimensionality. The Autoencoder and MLP share the same initial layers up to the reduced dimension layer, as shown in Figure 1.

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

We ran the simulator several times to generate initial sample set was used by the calibration framework to explore the relationship between the inputs and outputs. Additionally, to

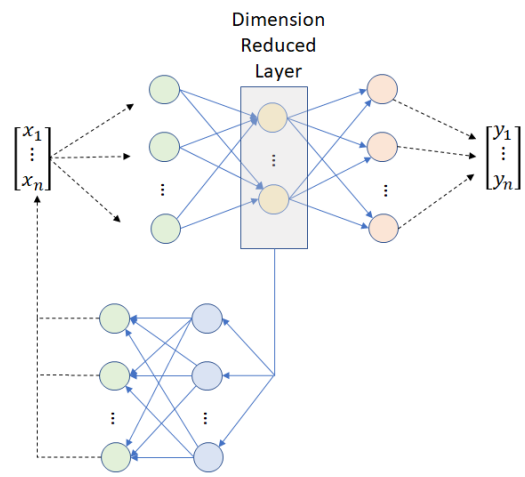


Fig. 1. Graphical Representation of the Combinatorial Neural Network for Calibration

quantify the discrepancies during training, the following loss functions were used:

1) The MLP portion of the architecture used 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 used the mean squared error function L and a quadratic penality cost P for producing predicted values outside of the original subspace bounds

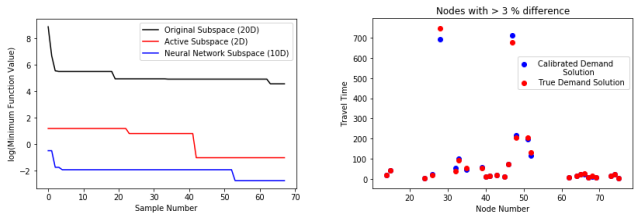
$$D\left[\phi(\theta)\right] = \max[0, \phi(\theta_i) - x_u]^2 + \max[0, x_l - \phi(\theta_i)]^2 \tag{2}$$

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

B. Empirical Results

We use Sioux-Falls [45], transportation model for our empirical results. This model consists of 24 traffic analysis zones and 24 intersections with 76 directional roads, or arcs. The network structure and input data provided by [46] 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. Random noise



(a) Objective Function (b) Calibrated vs True

Fig. 2. 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.

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.

Overall, the performance of the calibration using a deep neural network proved significant, see Figure 2(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 2(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.

IV. Deep Reinforcement Learning

Consider a problem of optimal design, e.g. an issue of a real-time management of a fleet. Once a system has been sufficiently calibrated to represent the real circumstances, we need to design policy $\pi : s \rightarrow a$ which maps the state of the system s to an optimal control action a. A typical way to design such a policy is to use a simulator combined with methods of either stochastic [47] or convex optimization [48] when the simulator has an analytical representation, such as partial differential equations. However, those simulators suffer from structural errors and parameters errors, corresponding to misspecified mathematical models or incorrect parameters of those models correspondingly [49]. Thus, once we start deploying a policy π learned from a simulator in the field, this policy might be far from being optimal [50]. At the same time, as we start using this policy, we start collecting data about the response of the travelers to this policy and how it impacts performance of the system, e.g. transit ridership or congestion patterns.

Now, the goal is to be able to use the observed data and update the policy to adjust for structural errors that were present in the simulator during the initial policy identification process. The framework of classical optimization and control approaches allows to incorporate the observed data by updating the simulator (parameters and mathematical models). This simulator-based process, which transforms the data into the models and its parameters, and subsequent simulation runs can be very expensive, both in terms of computing resources and modeler's time. We aim at developing a more viable approach that relies on reinforcement learning techniques so that observed field data can be used to update control policies. In other words, instead of having an expert that uses observed data to tweak the simulation model and then to retrain the policy, the new framework will automate this process and will make it more accessible and robust to inaccuracies present in simulators.

Our goal is to develop reinforcement learning techniques for finding optimal policies, that provides a viable alternative to more classical model predictive control (MPC) techniques [51]. The four basic components of an RL problem formulation are (i) observation (or current state) s, (ii) action a, (iii) transition $T(s^+; a, s)$ a probability of reaching next state s^+, given current state and action taken, and (iv) reward $R(\cdot)$. The goal of RL is to calculate an optimal policy $\pi : s \rightarrow a$ by maximizing the sum of future rewards. The classical dynamic programming approach calculates an action, given the state is via Bellman equation $Q(s,a) = R(s,a) + \gamma \sum_{s^+} T(s,a,s^+) \max_{a^+} Q(s^+, a^+)$. Here, function Q (a.k.a q-function) is the state-action value function, and γ is a discount factor for future rewards.

Model based approach relies on having a model for T and R functions, for example we can learn it from observed data $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{T}$ and then used the model to estimate value-function Q and then policy π. If there is an approximation error in the model, then we have compounding errors in he chain of estimations from one time step to another. Model based approach can be improved by using sequential learning of T, R and Q, e.g. assuming stochastic parameters and performing Bayesian update with each new observation or using warm start optimization, such as stochastic gradient descent. This is the key advantage or RL technique when compared to simulation optimization or model predictive approaches. The sequential learning allows to update the model as new measured data arrives.

The parametrization of the space of possible policies brings us to the problem of finding a deep learning representation of a possible control function.

We can skip the step of learning T and R functions and learn value function Q directly. That leads us to a model free approach known as Q - learning [52]. Referencing the "quality" of a certain action within the environment's current state, Q 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.

We learn Q function directly from observations $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{T}$. We start by generating some initial guess about Q, i.e. uniform $Q(s,a) = c$, than for each observation $\{s_t, a_t, r_t, s_{t+1}\}$ we calculate the error

$$\delta(s_t, a_t) = r_t + \gamma \max_{a^+} Q(s_{t+1}, a^+) - Q(s,a)$$

and then update the value of Q function

$$Q(s_t, a_t) = (1-\lambda)Q(s_t, a_t) + \alpha\delta(s_t, a_t)$$

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 a simulator continues to possess high-dimensionality concerns. However, recent advancements have allowed for the successful integration of reinforcement learning's Q-function with deep neural networks.[53] 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[54]

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

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.

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 3, 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 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

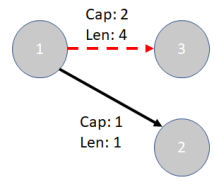


Fig. 3. Graphical Representation of the Example Transportation System

2) $0-2$ units of demand from node 1 to node 2 can be rerouted 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 $24-(t+1)$ The action 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 node 2 to the arc between node 1 and node 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 **??**,which implements the iterative Frank-Wolfe method to determine the traffic equilibrium flows and outputs total system travel time for the period. Since Q-learning seeks the maximum reward, we took the negative total system travel time.

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 4 compares the travel times by period between the original and adjusted demands.

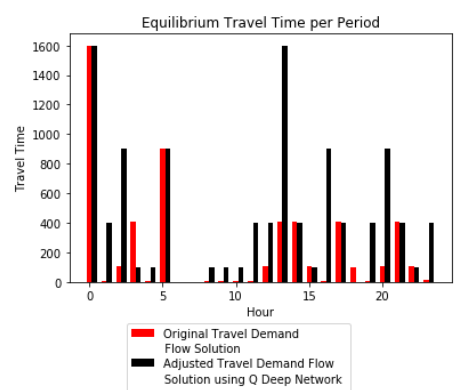


Fig. 4. Comparison of System Travel Time per Period

TABLE I
ADJUSTMENTS TO DEMANDS PER PERIOD USING DQN

Hour	Original Demand of $Arc_{1,2}$	DQN Adjusted Demand of $Arc_{1,2}$	DQN Adjusted Demand of $Arc_{1,3}$
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

V. 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 [55]. 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 [56]. 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 [52] who shows that Q-learning, which is an iterative scheme to learn optimal policies, does converge to optimal solution Q^* asymptotically. Littman et.el. [57] 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 [58]. Adaptive approaches that rely on meta-learning were shown to improve performance of reinforcement learning algorithms [59].

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 [60].

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. Fltterd, "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] 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.

[15] 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.

[16] 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.

[17] 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.

[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. Fltterd, "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. Fltterd, 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. Fltterd, 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] B. Zhang, M. Fu, and H. Yan, "A nonlinear neural network model of mixture of local principal component analysis: application to handwritten digits recognition," *Journal of the Pattern Recognition Society*, vol. 34, no. 2, pp. 203–214, Feb. 2001. [Online]. Available: https://ac.els-cdn.com/S0031320300000091/1-s2.0-S0031320300000091-main.pdf?_tid=c01e0bfe-09d3-11e8-ab8c-00000aab0f27&acdnat=1517766758_381d7cb5829ac2f9ab47cdf2258c803e

[44] V. N. Marivate, F. V. Nelwamodo, and T. Marwala, "Autoencoder, principal component analysis and support vector regression for data imputation," *arXiv preprint arXiv:0709.2506*, 2007.

[45] 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.

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

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

[48] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[49] C. Osorio and K. K. Selvam, "Simulation-based optimization: achieving computational efficiency through the use of multiple simulators," *Transportation Science*, vol. 51, no. 2, pp. 395–411, 2017.

[50] H. Kim, "Statistical modeling of simulation errors and their reduction via response surface techniques," Ph.D. dissertation, 2001.

[51] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 517–529, Apr. 2009.

[52] 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

[53] 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

[54] 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

[55] 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.

[56] 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

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

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

[59] 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.

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