Future Generation Computer Systems 135 (2022) 172–180

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Near-optimal responsive traffic engineering in software defined networks based on deep learning

Mohammed I. Salman [a,b,c,*], Bin Wang [a]

[a] Wright State University, Dayton, OH, USA
[b] Miami University, Oxford, OH, USA
[c] Anbar University, Ramadi, Anbar, Iraq

ARTICLE INFO

Article history:
Received 10 July 2021
Received in revised form 28 January 2022
Accepted 28 April 2022
Available online 6 May 2022

Keywords:
Traffic engineering
Network traffic control
Software defined networking
Oblivious routing
Optimization
Deep learning

ABSTRACT

The routing problem for traffic engineering can be solved using different techniques. For example, the problem can be formulated as a linear program (LP) or a mixed-integer linear program (MILP) that requires solving a complex optimization problem. Thus, this approach typically cannot be used for solving a large problem in real time. Alternatively, heuristic algorithms may be devised that, though fast, do not guarantee an optimal decision. This work proposes a novel design of a system that employs a deep learning model trained on optimal decisions to solve the routing problem. The model learns to adapt to traffic dynamics by updating the traffic split ratios to distribute traffic to a few paths between a source and a destination instead of frequently computing a single path for a source and destination pair. This solves the problem of network disturbance and traffic disruption. Specifically, we train two deep learning models: DNN (MLP), which is fully connected layers of neurons, and DNN (LSTM) that consists of a few layers of LSTM neural network and a dense layer. The two models are evaluated in a TE simulator. The system offers two important features of a good traffic engineering system: producing close to optimal traffic engineering results and responding to traffic dynamics in real time. We perform simulations on two topologies, the ATT North America topology, and a 4x4 grid topology. The results show that our proposed system can learn from optimal decisions to attain a responsive traffic engineering system.

1. Introduction

Recently, there have been many proposals that exploit Machine Learning (ML)/Deep Learning (DL) techniques to solve the routing problem in Traffic Engineering (TE) to minimize the network congestion or delay [1–12]. Regardless of the different optimization objectives, most past research treats the TE problem as finding a dynamic single path between any source–destination (SD) pair. It has been shown that multi-path routing, i.e., traffic splitting across different paths, is more advantageous than single-path routing, as we will show in Section 2. A dynamic optimal single path in a production network can cause additional overhead to network elements, i.e., switching or routing devices, due to the need for installing new paths whenever there is a change in the network traffic patterns. It has been shown that, under real network settings, traffic splittings between paths are relatively fast operation compared to installing new routes in the system that may take several minutes to update on many geo-distributed networking devices [13]. Besides, installing and deleting routes may introduce network disruption and stability problems. Thus, it is more advisable to choose static multi-path routing over dynamic single-path routing.

Considerably, the most prevalent solutions to the TE problem is to model it using LP or MILP. The use of such mathematical modeling can give an optimal solution, but requires significant amount of computation, and therefore is not responsive to real time requirements [2]. As a result, researchers have designed heuristic solutions to solve for the same objective function, a task that is non-trivial in many cases and does not guarantee an optimal or near-optimal solution [14,15].

ML/DL offers promising solutions to the TE problem. Researchers often use a Supervised Learning (SL) model that was trained on sub-optimal or local optimal data, e.g., training data obtained from standard routing protocols [1] such as the Open-Shortest Path First (OSPF) protocol. However, an ML model trained on such data cannot produce optimal or quasi-optimal routing decisions. This is the reason why researchers tend to use Reinforcement Learning (RL). However, reinforcement learning has two major problems that have not been addressed yet, proof of optimality and speed of convergence [16].

* Corresponding author at: Miami University, Oxford, OH, USA.
E-mail addresses: salmanmi@miamioh.edu (M.I. Salman), bin.wang@wright.edu (B. Wang).

https://doi.org/10.1016/j.future.2022.04.036
0167-739X/

In this work, we strive for a TE system with the following desirable characteristics:

- **Multipath TE** The TE system makes use of more than one path between each SD pair to take full advantage of network resources.
- **Optimality** The TE solution is optimal or quasi-optimal.
- **Responsiveness** The TE system must quickly adapt to changing traffic patterns while producing the optimal or quasi-optimal solutions.
- **Stability** The TE system must ensure stability by using a few precomputed paths or dynamic paths but with minimal changes to these paths in response to changing traffic patterns.

Deep Neural Networks (DNNs) can generalize from previously seen examples and process a new input instantaneously after the training process is finished. Unlike traditional Machine Learning (ML) techniques, DNN can perform feature engineering with the underlying learning system instead of selecting features manually. In addition, it is challenging to perform feature engineering for the routing problem as all the input features we use are of the same type, a quantity of demand that needs to be forwarded from a source to a destination.

To this end, we propose DNN models to learn traffic forwarding based on traffic splitting instead of learning optimal paths. We train the DNN to learn traffic routing based on traffic engineering decisions from a TE system known as RACKE+AD and described in [17], that has been shown superiority over other TE systems in terms of throughput, delay, and congestion. The RACKE+AD TE system forwards traffic based on routes computed using the Räcke's oblivious routing algorithm [18–20] and split traffic across these routes based on the average delay objective function, described in Section 2. Routes selected using the Räcke's model mitigate congestion that occurs when the TE system always selects the shortest path without being aware of network link capacity.

The routing problem is not limited to communication networks but can arise in many types of networks, such as transportation networks [21], and complex networks [22,23].

The contributions of this work are as follows:

- We propose two DL models, DNN (MLP) and DNN (LSTM), for the routing problem. The models learn traffic split ratios obtained from the optimal solutions as a result of solving the routing problems using LP. Furthermore, we test the trained models in a TE simulator and report the gap in throughput between the optimal LP solution and the solution we get from the trained models.
- We compare the performance of the two proposed DL models. The result confirms that the LSTM neural network performs better than MLP.
- We show the effectiveness of multi-path routing over single-path routing and their impact on network performance.

The rest of this paper is organized as follows. In Section 2, we describe the motivation behind our proposal with some early experimental results that show the performance of different TE systems. In Section 3, we discuss prior work with a focus on the field of traffic engineering. We describe our system in Section 4. In Section 5, we report and discuss our findings. Finally, in Section 6, we present the conclusions.

2. Motivation

To the best of our knowledge, most of the previously proposed ML/DL solutions for the routing problem involve a single path planning [1–8] and ignore the effect of multipath routing on network performance, i.e., the lack of modeling a TE system that splits traffic across many available paths between the source and destination nodes. In this section, we perform a series of simulations to show the benefit of splitting traffic across a few static paths between each SD pair over forwarding traffic on a single but dynamic optimal path between each SD pair.

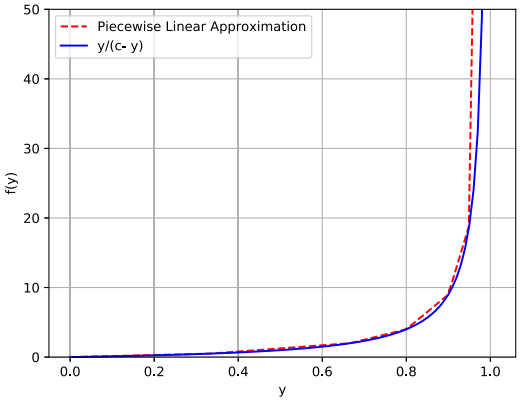


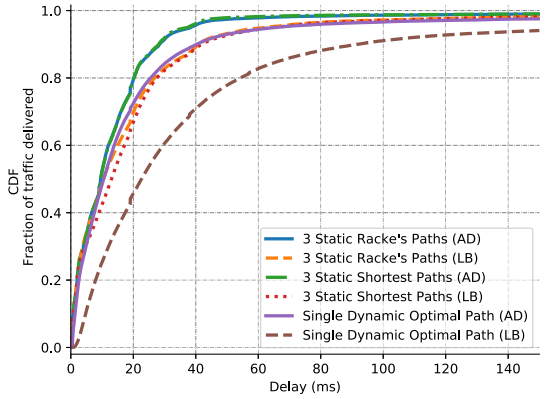
Fig. 1. Piece-wise linear approximation of the delay function.

Fig. 2. Abilene topology. *Source:* Regenerated from [25].

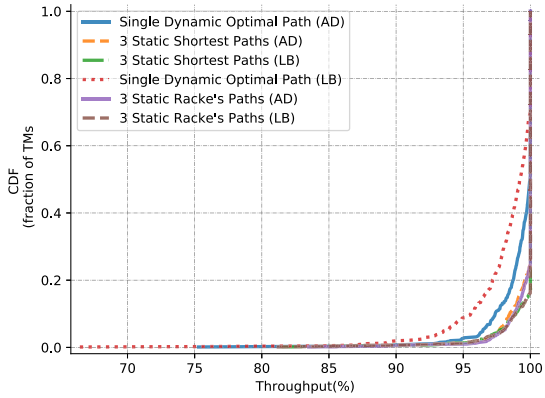
Simulation Setup. We compare six TE systems, four of which use three static paths between each SD pair. These four multi-path systems are the combinations of two path selection strategies and two traffic splitting objective functions. The two routing strategies are the k-shortest paths and paths extracted from the Räcke's oblivious routing algorithm. The two traffic splitting objective functions are Load Balance (LB), also known in the literature as the minimization of the maximum link utilization (MLU), and Average Delay (AD) objective function which is a piecewise linear approximation of the delay function depicted in Fig. 1. The other two TE systems are the optimal systems that use a single dynamic path with the same two traffic objective functions mentioned earlier. We conduct the experiment using the Abilene topology[1] as depicted in Fig. 2. The traffic matrices are generated using the gravity model [13,24].

Fig. 3 shows the cumulative distribution function of the six TE systems for 3 performance metrics: delay, throughput, and link congestion. The results confirm that TE systems with dynamic optimal single paths do not perform better than multi-path systems. For example, in Fig. 3(c), using 3 static paths over one single

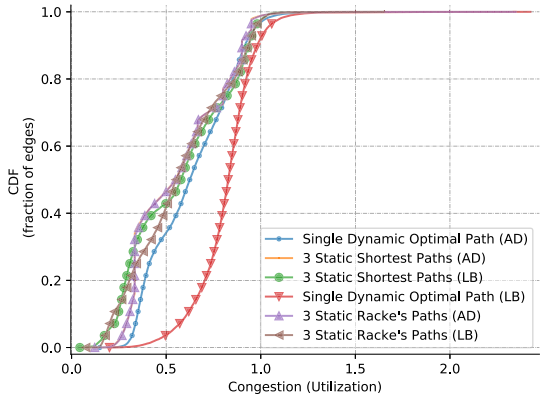
[1] Dataset available at: http://www.topology-zoo.org/dataset.html.



(a) CDF of latency



(b) CDF of throughput



(c) CDF of link congestion

Fig. 3. Three measured metrics for the Abilene topology for different routing strategy settings.

path enhanced the utilization of about 40% when the LB objective function was used. Thus, we propose a deep learning model that learns traffic splitting instead of learning the best single path to route traffic. Furthermore, we propose modeling a TE system with paths chosen based on the Räcke's oblivious routing model and traffic splitting based on the AD objective function. This traffic engineering approach achieves comparable performance in delay, throughput, and link utilization with respect to the optimal traffic engineering [17]. The AD objective function has shown better performance than MLU [17] due to the degraded performance of MLU under heavy traffic loads, which may limit the total throughput of the network [26]. The aforementioned TE system is referred to in the literature as *RACKE+AD,SALMAN202155*.

3. Related work

The usual approach to solve the routing problem is to use LP for multi-path routing strategy and MILP for single-path routing strategy. In general, both strategies are intractable since the runtime scales quadratically with the topology size [27,28]. Since network traffic is dynamic, a new allocation of flows has to be recomputed periodically and if routing configurations are not updated in time, the network will be underutilized.

Some works have exploited ML/DL solutions to solve the routing problem. The work of [5] proposed a sequence-to-sequence deep learning model to predict the forwarding path between each SD pair from historical forwarding experiences. They further used an attention mechanism [29] and beam search [30] to ensure the connectivity and a proper ordering of nodes. However, as mentioned earlier, relying on historical forwarding experiences does not guarantee an optimal solution. In [2], in an approach similar to our approach, they proposed a DL model trained on optimal decisions attained by solving MILP optimization problems to predict the optimal single path between every two nodes. Our approach is different from their approach in that we use multi-path routing instead of a single path, and the TE system we use to learn the model is also different from their system.

In [1,6,7] the authors proposed a decentralized deep learning system where each node has several DL models equal to the number of destinations in the network. This decentralized DL model works in a hop-by-hop fashion by predicting the next hop the packets should be forwarded to. All the DL models along the path collaborate to form the path of the packet. The drawback in this approach is that each node has to train as many models as the number of destinations. They tested their approach on a small topology, a 4×4 grid; however, only 12 actual nodes were used as edge routers. The other 4 intermediate nodes in the middle serve as forwarding nodes. In addition, they collected their data based on the OSPF routing protocol. OSPF routing protocol requires calculating the shortest path periodically, increasing CPU utilization, and disturbing network services. Furthermore, it has been shown in the literature that utilizing the shortest path may not help minimize the congestion due to many flows competing for the same highly utilized link [13,17]. Moreover, the work in [1,6,7] requires propagating the traffic patterns to all the models periodically, resulting in increased overhead.

In [31] the authors proposed a DL-based distributed routing system that can guarantee connectivity with the help of the link reversal theory. They have shown that even a small error in the DL model can result in routing loops/blackholes. However, similar to [1,6,7], their approach is distributed hop-by-hop routing that has difficulty in achieving optimality. Furthermore, they use the shortest path routing, which may degrade performance because many flows may compete for the same bottle-necked link. They also use the load balancing objective function, which has some performance issue under stressed traffic conditions [17,26]. In a similar approach in [3], they trained a separate model for each source and destination pair in the network. Furthermore, they use supervised learning with data generated from heuristic solutions, which does not guarantee optimality.

Recently, there have been breakthroughs in the field of AI by combining Deep Learning with Reinforcement Learning (RL), Deep Reinforcement Learning (DRL). One prominent example in [32] where the authors created an agent that performed different and challenging tasks of learning to play 49 games on the Atari 2600 platform. Another example in [33] where the agent

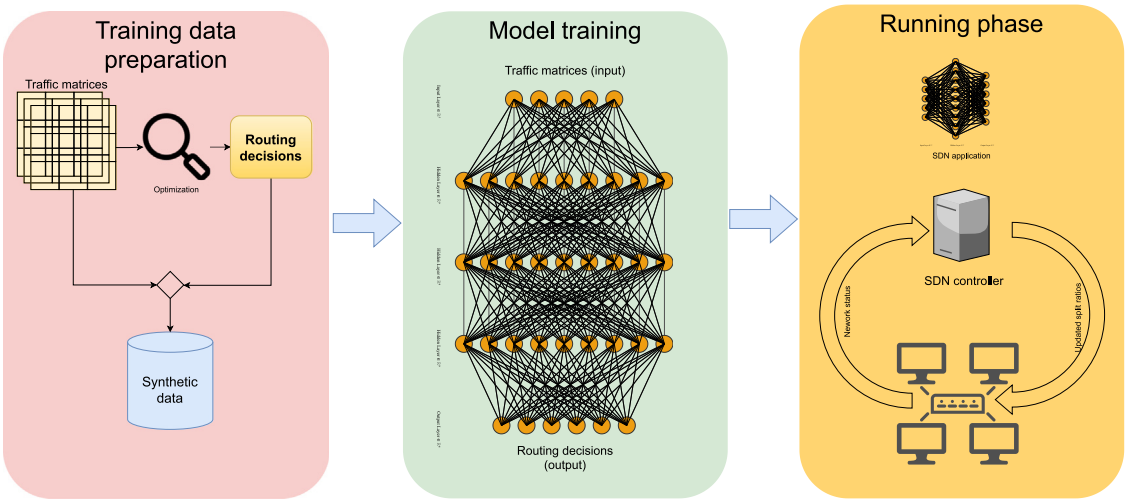


Fig. 4. A pipeline showing the steps of the proposed traffic engineering system that leverages Deep Learning.

learns to play the game of Go that has a gigantic search space. In DRL, an agent is responsible for observing which set of actions lead to better performance. The agent sends actions as input and receives observations and rewards as output. The reward serves as feedback to the learning algorithm. From the TE perspective, an input might be a set of selected routes as actions to forward traffic, and an output might be congestion status as observations and the maximum utilized link as a metric (or a reward). Some notable examples of RL/DRL for the routing problem can be found in [4,9–11]. There are two main drawbacks with deep reinforcement learning or reinforcement learning when it comes to traffic engineering: (1) it learns to route traffic based on its own experience, without using labeled data. Learning from historical decisions, however, does not ensure the optimal or near-optimal solutions. The algorithm may provide solutions that are locally optimal. According to a study in [16], proof of convergence is not thoroughly and convincingly addressed yet. To ensure the optimal solution, the model may have to rely on optimally labeled data. (2) Another issue with RL is the speed of convergence. According to the same study in [16], further investigation is required to provide the bounded delay of the routing decision made by the RL-based routing algorithm.

The paramount difference between previous work and our work is that the routing problem is usually treated as selecting paths to send packets from sources to destinations in previous work. In contrast, our work treats the problem as splitting traffic among a few paths that are preselected for each SD pair.

4. System description

This section describes our proposed model that learns near-optimal traffic split ratios that will be used to route traffic flows for better delay, throughput, and resource utilization. Fig. 4 shows the three main steps of the proposed system.

Please note that the first two steps, training data preparation and model training, happen offline, then the resulting model can be applied online through the third step, the running phase.

4.1. Training data preparation

Training data are generated, in an offline mode, by solving tens of thousands of problem instances in parallel using a Linear Program (LP). One LP instance takes the topology information and TM as input. It produces the corresponding routing strategy, i.e., split ratios over a fixed set of routes as output. We use the Gurobi optimization software [34] to solve these problem formulations in parallel. To output a valid routing prediction, data should not be scaled or normalized on a per column basis. Instead, the whole dataset should be normalized or scaled altogether as there are dependencies between individual columns in the dataset. Alternatively, data can be generated on a scaled topology, i.e., link capacities are scaled in the range [0, 1]. This is necessary to match the output of activation functions (for example, ReLU and Sigmoid).

4.2. Model training

We train two deep learning models, DNN (MLP) and DNN (LSTM), that, once trained, can find the routing decision instantaneously. They learn mapping traffic matrices to their corresponding near-optimal traffic split ratios, with routes selected statically using the Räcke's oblivious routing model [19,20]. To ensure unbiased evaluation, each traffic matrix (TM) is generated independently from a distribution of a sparsified gravity model. The split ratios are calculated based on the piecewise linear approximation (PLA) [17,35] of the average delay objective function [36]. One of the motivations for considering demands only as input features is that they directly relate to network status. When a new flow arrives, some links will be affected. Likewise, when an existing flow terminates, some links' loads will be reduced. The traffic matrices can be easily collected due to SDN technology. The neural network models are trained using a dataset generated as described in Section 4.1 by solving many optimization problems using LP. LP is an optimal method to solve the routing problem but may not be efficient, especially for a large network. Thus, in this work, we try to utilize the output we get from LP to train a neural network model to do the job instead. In our case, the routing strategy is to obtain only split ratios of traffic per each SD pair over a limited number of static routes. These routes are calculated based on the Racke's oblivious routing model that has shown superiority [13,17] over the shortest path-based techniques [37]. Similar to the first step in Section 4.1, this step is also done in an offline mode.

4.3. Running phase

When the training is completed, the SDN controller can use the model to provide switches with the new split ratios periodically. The frequency at which the routing decision is updated is up to the network operator. The real-time traffic matrix must be collected by the SDN controller and fed to the model to decide on the

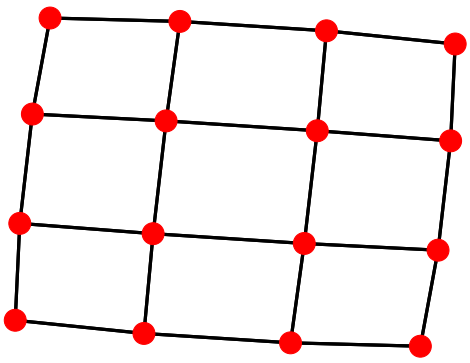


Fig. 5. A 4 *x* 4 grid topology used in evaluation.

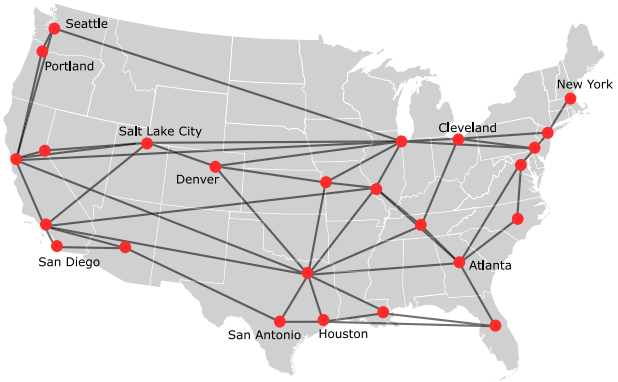


Fig. 6. The ATT North America topology.
Source: Regenerated from [25]

new ratios. The evaluation results have shown that this model is accurate enough to achieve the near-optimal performance. Using this approach, the SDN controller does not need to install new routes on switching devices; instead, the same routes are used. Traffic is adapted to changing traffic patterns by changing the split ratios only. In a production network, updating split ratios is a relatively cheap operation compared to the operation of updating routes on network elements [13].

The quantity of the split ratios might be quite large for large networks. A network of N nodes with 4 selected routes per each SD pair (assuming 4 routes per SD pair are available) will lead to a total of $N * (N - 1) * 4$ split ratios. It has been shown that using only a few routes per SD pair can give a performance that is only a few percent away from the optimal [13,17,38]. Because the running phase relies on the trained model, this phase is applied online. Thus, the SDN controller can query the model and get the routing configuration in real-time. This process of querying the trained model is much faster compared to solving an optimization problem.

5. Performance evaluation

This section evaluates the performance of the two proposed deep learning models using two topologies, a 4 *x* 4 grid topology (Fig. 5) and ATT North America topology (Fig. 6). The information of these two topologies is given in Table 1. We trained two models, DNN (LSTM) and DNN (MLP), for each topology.

Table 1
Network topologies used in the evaluation.

Topology	Nodes	Directed links
4 x 4 grid	16	48
ATT North America	25	114

5.1. Evaluation setup

We conducted model training on a remote computer with an Intel Xeon 2.20 GHz CPU, 16 GB RAM, and Tesla V100-SXM2-16 GB GPU. Due to the large size of the output (routing decision) that needs to be predicted by the neural network, we restricted our evaluation on two backbone topologies, a 4 *x* 4 grid topology and the ATT North America topology. The length of the input vector represents the number of demands between all SD pairs. Thus the input vector length is 240 and 600 for the 4 *x* 4 grid topology and the ATT North America topology, respectively. The length of the output vector that needs to be predicted depends on the input vector, i.e., the number of demands and the number of paths used between each SD pair. Thus, with only 3 paths used between each SD pair, the length of the output vector is 720 and 1800 for the two topologies, respectively. However, although 3 paths are allocated for each SD pair, some paths are not used at all as this depends on the objective function being used. We use grid search to decide on the number of neurons and number of layers in the DNN and other hyperparameters. We tried different configurations regarding the learning rate, drop rate, and the optimization algorithm. The rest of the used hyperparameters are depicted in Table 2. We checked the performance of the model for each configuration by testing it on a validation set, and selected the model with the least Mean Absolute Error (MAE). The MAE was also used to compare the performance of the two mentioned DNN architectures. For DNN (MLP), the best performance is achieved with two layers. The first layer has a number of neurons 25% larger than the length of the input vector, and the second layer has a number of neurons equal to the length of the output vector. For the DNN (LSTM), 2 layers of LSTM cells with one last dense layer were used.

5.2. Evaluation

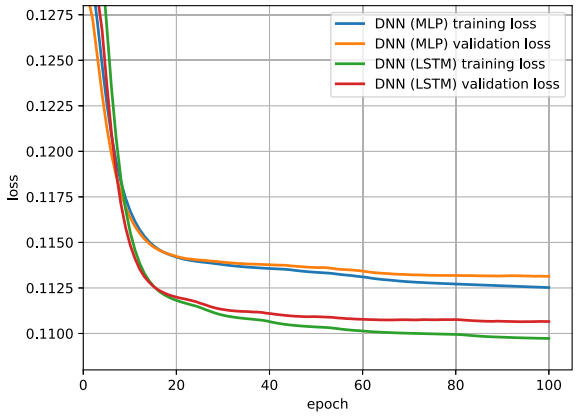
In this section, the performance of our two proposed deep learning models is evaluated. We evaluated the performance in two stages. First, we show how the models are learning traffic splitting with decreasing model training error over time (Figs. 7(a) and 8(a)). Second, we perform model inference (Figs. 7(b) and 8(b)) by integrating the best model obtained in stage 1 into a TE simulator from [17]. The response time for each approach is reported in Table 4. Using the TE simulator, we show how close the throughput from the DL model to the throughput obtained from solving the LP in two cases, RACKE+AD and optimal. We define the optimal solution as using all the paths between all the source–destination pairs in the network with the AD objective function described in Section 2. We refer to this optimal solution as OPTIMAL (AD).

5.2.1. Model training

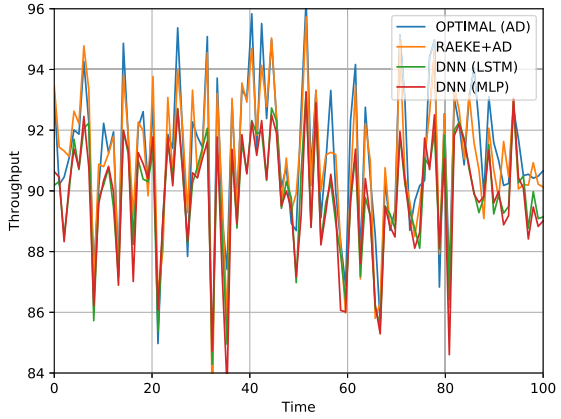
As can be seen in Figs. 7(a) and 8(a), both models are able to learn the splitting ratios for both topologies. The model that uses LSTM results in a slightly better prediction performance. This could be attributed to the fact that LSTM has memory and a much more complex architecture than MLP. However, the results show slight overfitting to the training data, which can be mitigated using the deep learning regularization techniques. Both models took less than one hour to finish training, making them feasible to retrain in case of topology change.

Table 2
Hyperparameters used for training DNN (LSTM) and DNN (MLP) for two topologies. The best achieved model is with parameters highlighted in a bold font.

	4 x 4 grid		ATT North America	
	DNN (MLP)	DNN (LSTM)	DNN (MLP)	DNN (LSTM)
Learning rate α	0.01, **0.001**	0.01, **0.001**	0.01, **0.001**	0.01, **0.001**
Dropout rate	0, **0.1**, 0.2	0, 0.1, **0.2**	0, **0.1**, 0.2	0, 0.1, **0.2**
Optimizer	**Adam**, RMSProp	**Adam**, RMSProp	**Adam**, RMSProp	**Adam**, RMSProp
Layers information:				
1 hidden layer	[255]	[65]	[450]	[80]
	[240]	[85]	[600]	[100]
	[300]	[95]	[750]	[120]
	[360]	[115]	[900]	[140]
2 hidden layers	[240, 720]	[65, 95]	[600, 1800]	[80, 120]
	[300, 720]	**[85, 85]**	**[750, 1800]**	[90, 110]
	[360, 720]	[95, 65]	[900, 1800]	**[100, 100]**
				[110, 90]

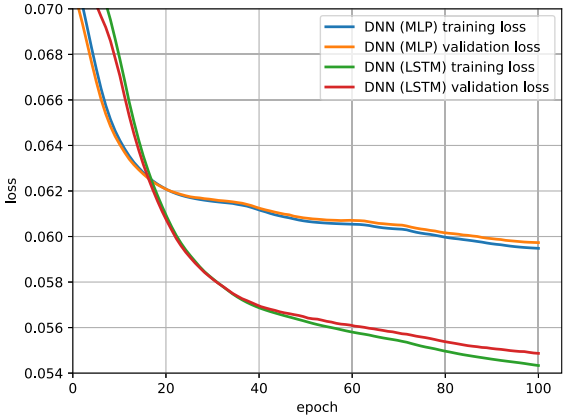


(a) The mean absolute error in DNN (MLP) model and DNN (LSTM) model for 4 x 4 grid topology.

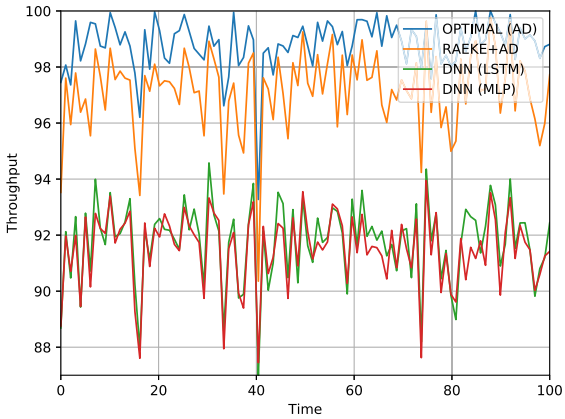


(b) Throughput for 4 x 4 grid topology.

Fig. 7. (a) Training errors for models; (b) network throughput achieved using the two models for 4 × 4 grid topology.



(a) The mean absolute error in DNN (MLP) model and DNN (LSTM) model for ATT North America topology.



(b) Throughput for ATT North America topology.

Fig. 8. (a) Training errors for models; (b) network throughput achieved using the two models for ATT North America topology.

Table 3
Overall average of network throughput.

Topology	Algorithm	Mean
4 x 4 grid	DNN (MLP)	89.836
	DNN (LSTM)	89.992
	RACKE+AD	90.990
	OPTIMAL (AD)	91.177
ATT North America	DNN (MLP)	91.506
	DNN (LSTM)	91.771
	RACKE+AD	97.136
	OPTIMAL (AD)	98.807

5.2.2. Model inference

We would like to see how the learned models from the first stage perform in the simulator. We integrated the two proposed DL models into the TE simulator to evaluate how the prediction performs against the original near-optimal solution obtained by the RACKE+AD TE system and the optimal solution. As shown in Figs. 7(b) and 8(b), network performance is evaluated for the two DL models, RACKE+AD TE, and the optimal solution, using the throughput metric. Because of the consequent overlapping between lines in Fig. 7(b), the results in Figs. 7(b) and 8(b) are aggregated in Table 3. The throughput on average, as can be seen in Table 3, is about 1% and 6% away from the RACKE+AD throughput for the 4 *x* 4 grid and the ATT North America topologies, respectively. The gap in the throughput between the learned models and the RACKE+AD system on the 4 *x* 4 grid topology is much smaller than the gap on the ATT North America topology. This is due to the much smaller size of the predicted splitting ratios vector of the 4 *x* 4 grid compared to the output size in the ATT North America topology which is 720 to 1800. In the simulator, although it is not the case in practice, we assumed that the RACKE+AD TE system could find the optimal solution instantaneously without accounting for the time required to find the solution each time the TM is updated. Thus, we expect the learned DL models to perform much better in a real production network than the solution obtained by solving LP, RACKE+AD and OPTIMAL (AD). We compare our results mainly with RACKE+AD because our two DL models are trained on data obtained from RACKE+AD. However, the result for the optimal model is also provided as a reference. RACKE+AD was 0.187% and 1.671% away from the optimal solution in 4 *x* 4 grid and ATT North America topologies, respectively. It should be noted that although we provided the optimal solution in terms of performance (Figs. 7(b) and 8(b)) and time taken to find the solution (Table 4), in practice, it is difficult to apply the optimal solution due to three reasons. First, the optimal solution uses all the available routes in the system which introduces routes oscillation problem [13,17]. Second, switching devices typically have limited TCAM memory and it is difficult to store all available paths in that limited memory. Third, due to the large number of variables and constraints in the LP of the optimal solution, the optimal solution is difficult to be realized in real-time as can be seen in Table 4.

5.2.3. Responsiveness

Table 4 indicates that the two DL models require a remarkably less time in processing the input than the LP solutions, RACKE+AD and OPTIMAL (AD). The numbers in Table 4 (highlighted in a bold font) represent the total time in seconds an approach took to process the inputs of 100 traffic matrices.

For 4 *x* 4 the grid topology, the two DL models process the input 80.92%–83.48% faster than RACKE+AD while it is 93.38%–94.27% faster than OPTIMAL (AD). On the other hand, for the ATT North America topology, the two DL models process the input −77.99%–81.39% faster than RACKE+AD and 98.91%–99.07% faster than OPTIMAL (AD). As there is a trade-off between performance and responsiveness, we expect our learned approach to give better than or comparable performance of that of LP as network demands are dynamic in nature and frequent update is required.

6. Conclusions

We have presented the design and evaluation of a new DL-based traffic engineering system for a software-defined network. The new system achieves two important characteristics of a good TE system, closeness to optimality and fast responsiveness. The fast responsiveness is naturally achieved due to the instantaneous inference of ML models. The closeness to optimality is achieved by training DL models on optimal solutions calculated beforehand using linear programming mathematical modeling. The proposed system also achieves an important feature which is stability of network operation. The stability is achieved by changing traffic split ratios as needed rather than installing/deleting new/old routes in the network in response to the change in traffic patterns. The main difference between our work and previous work is how we characterize the input and the output of the deep learning model. We have shown that relying on traffic demand pattern only, as features, can give a good performance due to the direct relationship between demands and routing of demands over multiple paths. The output of the DL models is characterized by the traffic split ratios, as opposed to paths between SD pairs in previous work. However, the proposed system has shown an optimality gap with the ATT North America topology that is larger than the optimality gap with the 4 *x* 4 grid topology due to the much larger output for the ATT North American topology. The optimality gap is relatively high only for large networks due to the increased size in the output vector that needs to be predicted. However, during the simulations, we assumed that the linear program could make an instantaneous decision. While it actually takes several seconds to find a solution for a single TM in a production network. For future work, we would like to mitigate the gap in optimality in two ways: (1) by investigating different DNN architectures and (2) by training many DL models where each model is in charge of predicting a small part of the large output.

CRediT authorship contribution statement

Mohammed I. Salman: Conceptualization, Methodology, Software, Data curation, Writing – original draft, Visualization. **Bin Wang:** Validation, Formal analysis, Writing – review & editing, Project administration, Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and suggestions. We would like to thank the staff of the Ohio Supercomputer Center for their support by providing the computing facility during this study and installing the Gurobi software and license on their system based on our request.

Table 4
Total response time for 100 traffic matrices (in seconds). $\Delta 1$ is the percentage difference between the current model and RACKE+AD. $\Delta 2$ is the percentage difference between the current model and the optimal solution.

Algorithm	4 × 4 grid	($\Delta 1$)	($\Delta 2$)	ATT N. A.	($\Delta 1$)	($\Delta 2$)
DNN (MLP)	**20.85**	(−83.48%)	(−94.27%)	**25.48**	(−81.39%)	(−99.07%)
DNN (LSTM)	**24.09**	(−80.92%)	(−93.38%)	**30.134**	(−77.99%)	(−98.91%)
RACKE+AD	**126.26**	(0.0%)	(−65.32%)	**136.94**	(0.0%)	(−95.04%)
OPTIMAL (AD)	**364.10**	(+188.37%)	(0.0%)	**2765.52**	(+1919.51%)	(0.0%)

References

[1] Z.M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, K. Mizutani, State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2432–2455.

[2] Joao Reis, Miguel Rocha, Truong Khoa Phan, David Griffin, Franck Le, Miguel Rio, Deep neural networks for network routing, in: 2019 International Joint Conference on Neural Networks, IJCNN, 2019, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2019.8851733.

[3] Yanjun Li, Xiaobo Li, Yoshie Osamu, Traffic engineering framework with machine learning based meta-layer in software-defined networks, in: 2014 4th IEEE International Conference on Network Infrastructure and Digital Content, 2014, pp. 121–125, http://dx.doi.org/10.1109/ICNIDC.2014.7000278.

[4] Junjie Zhang, Minghao Ye, Zehua Guo, Chen-Yu Yen, H. Jonathan Chao, CFR-RL: Traffic engineering with reinforcement learning in SDN, IEEE J. Sel. Areas Commun. 38 (10) (2020) 2249–2259, http://dx.doi.org/10.1109/JSAC.2020.3000371.

[5] Yuan Zuo, Yulei Wu, Geyong Min, Laizhong Cui, Learning-based network path planning for traffic engineering, Future Gener. Comput. Syst. 92 (2019) 59–67, http://dx.doi.org/10.1016/j.future.2018.09.043.

[6] N. Kato, Z.M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, K. Mizutani, The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective, IEEE Wirel. Commun. 24 (3) (2017) 146–153.

[7] B. Mao, Z.M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, K. Mizutani, Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning, IEEE Trans. Comput. 66 (11) (2017) 1946–1960.

[8] Bomin Mao, Fengxiao Tang, Zubair Md Fadlullah, Nei Kato, An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems, IEEE Trans. Emerg. Top. Comput. (2019) 1, http://dx.doi.org/10.1109/TETC.2019.2899407.

[9] Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés-Mulero, David Meyer, Sharon Barkai, Mike J. Hibbett, Giovani Estrada, Khaldun Ma'ruf, Florin Coras, Vina Ermagan, Hugo Latapie, Chris Cassar, John Evans, Fabio Maino, Jean Walrand, Albert Cabellos, Knowledge-defined networking, SIGCOMM Comput. Commun. Rev. 47 (3) (2017) 2–10, http://dx.doi.org/10.1145/3138808.3138810.

[10] Jose Suarez-Varela, Albert Mestres, Junlin Yu, Li Kuang, Haoyu Feng, Pere Barlet-Ros, Albert Cabellos-Aparicio, Feature engineering for deep reinforcement learning based routing, in: ICC 2019 - 2019 IEEE International Conference on Communications, ICC, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICC.2019.8761276.

[11] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, Dejun Yang, Experience-driven networking: A deep reinforcement learning based approach, in: IEEE INFOCOM'2018, 2018.

[12] Paulo Alexandre Regis, Suman Bhunia, Amar Nath Patra, Shamik Sengupta, Deep-learning assisted cross-layer routing in multi-hop wireless network, in: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), 2021, pp. 35–39, http://dx.doi.org/10.1109/WF-IoT51360.2021.9595521.

[13] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, Robert Soulé, Semi-oblivious traffic engineering: The road not taken, in: Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation, in: NSDI'18, USENIX Association, USA, 2018, pp. 157–170.

[14] M.M. Tajiki, B. Akbari, M. Shojafar, S.H. Ghasemi, M.L. Barazandeh, N. Mokari, L. Chiaraviglio, M. Zink, CECT: computationally efficient congestion-avoidance and traffic engineering in software-defined cloud data centers, Cluster Comput. 21 (4) (2018) 1881–1897, http://dx.doi.org/10.1007/s10586-018-2815-6.

[15] W. Quan, N. Cheng, M. Qin, H. Zhang, H.A. Chan, X. Shen, Adaptive transmission control for software defined vehicular networks, IEEE Wirel. Commun. Lett. 8 (3) (2019) 653–656, http://dx.doi.org/10.1109/LWC.2018.2879514.

[16] Zoubir Mammeri, Reinforcement learning based routing in networks: Review and classification of approaches, IEEE Access 7 (2019) 55916–55950, http://dx.doi.org/10.1109/ACCESS.2019.2913776.

[17] Mohammed I. Salman, Bin Wang, Boosting performance for software defined networks from traffic engineering perspective, Comput. Commun. 167 (2021) 55–62, http://dx.doi.org/10.1016/j.comcom.2020.12.018.

[18] H. Räcke, Minimizing congestion in general networks, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings, 2002, pp. 43–52.

[19] Harald Räcke, Optimal hierarchical decompositions for congestion minimization in networks, in: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, in: STOC '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 255–264, http://dx.doi.org/10.1145/1374376.1374415.

[20] Philipp Czerner, Harald Räcke, Compact oblivious routing in weighted graphs, 2020.

[21] Roberta Di Pace, A traffic control framework for urban networks based on within-day dynamic traffic flow models, Transportmetrica A 16 (2) (2020) 234–269, http://dx.doi.org/10.1080/23249935.2019.1692957.

[22] Henrique F. de Arruda, Filipi N. Silva, Luciano da F. Costa, Diego R. Amancio, Knowledge acquisition: A complex networks approach, Inform. Sci. 421 (2017) 154–166, http://dx.doi.org/10.1016/j.ins.2017.08.091.

[23] Lucas Guerreiro, Filipi N. Silva, Diego R. Amancio, A comparative analysis of knowledge acquisition performance in complex networks, Inform. Sci. 555 (2021) 46–57, http://dx.doi.org/10.1016/j.ins.2020.12.060.

[24] D. Applegate, E. Cohen, Making routing robust to changing traffic demands: Algorithms and evaluation, IEEE/ACM Trans. Netw. 14 (6) (2006) 1193–1206, http://dx.doi.org/10.1109/TNET.2006.886296.

[25] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The internet Topology Zoo, IEEE J. Sel. Areas Commun. 29 (9) (2011) 1765–1775, http://dx.doi.org/10.1109/JSAC.2011.111002.

[26] C. Zhang, S. Zhang, Y. Wang, W. Li, B. Jin, R. K. P. Mok, Q. Li, H. Xu, Scalable traffic engineering for higher throughput in heavily-loaded software defined networks, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–7, http://dx.doi.org/10.1109/NOMS47738.2020.9110259.

[27] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, Amin Vahdat, B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, in: SIGCOMM '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 74–87, http://dx.doi.org/10.1145/3230543.3230545.

[28] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, Peter Bailis, Contracting wide-area network topologies to solve flow problems quickly, in: 18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 21, USENIX Association, 2021, pp. 175–200.

[29] Thang Luong, Hieu Pham, Christopher D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1412–1421, http://dx.doi.org/10.18653/v1/D15-1166.

[30] Graham Neubig, Neural machine translation and sequence-to-sequence models: A tutorial, 2017, CoRR arXiv:1703.01619.

[31] Shihan Xiao, Haiyan Mao, Bo Wu, Wenjie Liu, Fenglin Li, Neural packet routing, in: Proceedings of the Workshop on Network Meets AI & ML, in: NetAI '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 28–34, http://dx.doi.org/10.1145/3405671.3405813.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, http://dx.doi.org/10.1038/nature14236.

[33] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine

Leach, Koray Kavukcuoglu, Thore Graepel, Demis Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489, http://dx.doi.org/10.1038/nature16961.

[34] Gurobi Optimization, LLC, Gurobi optimizer reference manual, 2020, URL http://www.gurobi.com.

[35] X. Liu, S. Mohanraj, M. Pióro, D. Medhi, Multipath routing from a traffic engineering perspective: How beneficial is it? in: 2014 IEEE 22nd International Conference on Network Protocols, 2014, pp. 143–154, http://dx.doi.org/10.1109/ICNP.2014.34.

[36] L. Fratta, M. Gerla, L. Kleinrock, The flow deviation method: An approach to store-and-forward communication network design, Networks 3 (2) (1973) 97–133, http://dx.doi.org/10.1002/net.3230030202, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230030202.

[37] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, Roger Wattenhofer, Achieving high utilization with software-driven WAN, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, in: SIGCOMM '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 15–26, http://dx.doi.org/10.1145/2486001.2486012.

[38] Victor Heorhiadi, Michael K. Reiter, Vyas Sekar, Simplifying software-defined network optimization using SOL, in: Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, in: NSDI'16, USENIX Association, USA, 2016, pp. 223–237.

Mohammed I. Salman is a Ph.D. student at the College of Engineering and Computer Science, Wright State University, Dayton, Ohio USA. He received his master's degree from Anbar University in 2013. His research covers network design problems and network optimization.

Prof. Bin Wang received his Ph.D. in electrical and computer engineering from the Ohio State University in 2000. He is a professor at the Department of Computer Science and Engineering, Wright State University, Dayton, Ohio USA. He received the US Department of Energy Early Career Award in 2003. His research has been supported by numerous federal agencies, including the National Science Foundation, the Department of Energy, and the DoD.