DeepConf: Automating Data Center Network Topologies Management with Machine Learning

Saim Salman[*], **Theophilus Benson**[*], **Asim Kadav**[‡]
Brown University[*] NEC Labs[‡]

Abstract

Recent work on improving performance and efficiency of data center networks has focused on applying domain specific properties of the workload or hardware that requires human expertise and may not scale to large or complex networks. In this paper, we argue that many data center networking techniques, e.g., routing, topology augmentation, energy savings, with diverse goals share design and architectural similarities. We develop a framework, DeepConf, that learns general intermediate representations of network topologies to solve these different tasks. To illustrate the strength of our approach, we evaluate a DeepConf-agent that tackles the data center topology augmentation problem. Our initial results are promising — DeepConf performs comparably to the optimal solution.

1 Introduction

Data center networks (DCN) are a crucial and important part of the Internet's ecosystem. Motivated by the importance of these networks, the networking community has explored techniques for improving and managing the performance of data center networks by: (1) designing better routing or traffic engineering algorithms [5, 6, 10, 8], (2) enriching the fixed topology with a limited number of flexible links [23, 9, 14, 13], and (3) removing corrupted and underutilized links from the topology [15, 11, 24].

Regardless of the approach, these topology-oriented techniques have three things in common: (1) Each is formalized as an optimization problem with a corresponding Linear Program (LP) or Integer Linear Program (ILP) solution. (2) Due to the impracticality of scalably solving these optimizations, greedy heuristics are employed to create approximate solutions. (3) Most of these heuristics do not generalize because they are intricately tied to the high-level application patterns and technological constraints. In particular, determining the optimal routes, the optimal location to add augmenting links, or the optimal set of links to remove under diverse and rapidly evolving conditions is challenging (even NP-hard) [23, 14, 13, 9]. Existing domain-specific heuristics provide suboptimal performance and are often limited to specific scenarios.

In this paper, we present a reinforcement learning based approach called DeepConf, that simplifies the process of designing ML models for a broad range of DCN topology problems and eliminates the challenges associated with efficiently training new models.

DeepConf uses a convolutional network using network data, e.g., traffic matrix, to generate an intermediate representation of the network's state that enables learning a broad range of data center problems. Moreover, while labeled production data crucial for ML is unavailable, empirical studies [16] show that modern data center traffic is highly predictable and thus amenable to offline learning with network simulators and historical traces. The proposed DeepConf framework provides a predefined RL model with the intermediate representation, a specific design for configuring this model to address different problems, an optimized simulator to enable efficient learning, and a Software-defined networking (SDN) platform for capturing network data and reconfiguring the network. In this paper, we make the following contributions: We present a novel RL-based SDN architecture for developing and training deep ML models for a broad range of DCN tasks. We implemented a DeepConf-agent which tackles the topology augmentation problem and evaluated it on representative topologies [12, 3] and traces [2], showing that our approach performs comparable to optimal.

2 Related Work

Our work is motivated by the recent success of applying machine learning and RL algorithms to computer games and robotic planning [19, 20, 21]. The most closely related works [7, 22] apply RL to Internet service provider (ISP) networks. Unlike these approaches, DeepConf focuses on data center networks which have significantly larger scale and higher velocity than traditional ISP networks. Additionally, while these approaches [7, 22] focus on network routing, DeepConf tackles the topology augmentation problem and explores the use of deep networks as function approximators for RL. Existing applications of machine learning to data centers focus on improving cluster scheduling [18] and more recently by Google to optimize Power Usage Effectiveness (PUE) [1]. Instead our system focuses on data center management operations of toplogy management and configuration problems.

3 Design

DeepConf Agents In defining each new DeepConf-agent, there are four main functions that a developer must specify: state space, action space, learning model, and reward function. The action space and reward are both specific to the management task being performed and are, in turn, unique to the agent. Respectively, they express the set of actions an agent can take during each step and the reward for the actions taken. The state space and learning models are more general and can be shared and reused across different DeepConf-agents. This is because of the fundamental similarities shared between the data center management problems (they all solve a very similar ILPs), and because the agents are specifically designed for data centers.

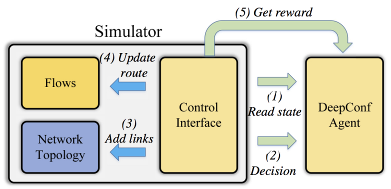
State Space: In general, the state space consists of two types of data – each reflecting the state of the network at a given point in time. First, the general network state that all DeepConf-agents require: the network's traffic matrix (TM) which contains information on the flows which were executed during the last t seconds of the simulation. Second, a DeepConf-agent specific state-space that captures the impact of the actions on the network. For example, for the topology augmentation problem, this would be the network topology – note, the actions change the topology. Whereas for the traffic engineering problem, this would be a mapping of flows to paths – note, the actions change a flow's routes.

Learning Model: Our learning model utilizes a Convolutional Neural Network (CNN) to compute policy decisions. The exact model for a DeepConf-agent depends on the number of state-spaces used as input. In general, the model will have as many CNN-blocks as there are state spaces – one CNN-block for each state space. The output of these blocks are concatenated together and input into two fully connected layers, followed by a softmax output layer. For example, for the topology-augmentation problem, as observed in Figure 1b, our DeepConf-agent has two CNN blocks to operate on both the topology and the TM states spaces in parallel. This model allows for the lower CNN layers to perform feature extraction on the input spatial data, and for the fully connected layers to assemble these features in a meaningful way.

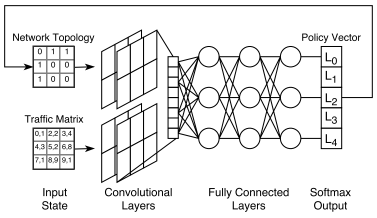
3.1 Network Training

To train a DeepConf-agent, we run the agent against a network simulator. The interaction between the simulator and the DeepConf-agent can be described as follows (Figure 1a): (1) The DeepConf-agent receives state s_t from the simulator at training step t. (2) The DeepConf-agent uses the state information to make a policy decision about the network and returns the selected actions to the simulator. For topology augmentation problem's DeepConf-agent, called the Augmentation-Agent, the actions are the set of K optical links to activate. (3) If the topology changes, the simulator re-computes the paths for the active flows. (4) The simulator executes the flows for x seconds. (5) The simulator returns the reward r_t and state s_{t+1} to the DeepConf-agent, and the process restarts.

Initialization During the initial training phase, we force the model to explore the environment by randomizing the selection process — the probability that an action is picked corresponds to the



(a) DeepConf-agent model training.



(b) The CNN model utilized by the DeepConf-agent.

Figure 1: DeepConfig Model Architecture

Input-Space	CNN-1	CNN-2	Fully-Connected	Value Network	Policy-Network
Traffic Matrix (NxDxDx2)	Nx5x5x(D*2)	Nx2x2x(D*4)	Nx256	Nx256x5	Nx256xO
Topology (NxDxD)	Nx5x5x(D*2)	Nx2x2x(D*4)			

Table 1: Model Configuration: As shown in Figure 1b, each input space has a set of CNN blocks which is fed into shared fully-connected layer, value network and policy-network.

value of the index which represents the action. For instance, with the Augmentation-Agent, link i has probability w_i of being selected. As the model becomes more familiar with the states and corresponding values, the model will better formulate its policy decision. At this point, the model will associate a higher probability with the links it believes to have a higher reward, which will cause these links to be selected at a higher frequency. This methodology allows for the model to *reinforce* its decisions about links, while the randomization helps the model avoid local-minima.

Learning Optimization: To improve the efficiency of learning, the RL agent maintains a log containing the state, policy decision, and corresponding reward. The RL agent performs *experience replay* after n simulation steps. The agent is trained using Adam stochastic optimization [17], with an initial learning rate of 10^{-4} and a learning rate decay factor of 0.95. We found that a smaller learning rate and low decay helped the model better explore the environment and form a more optimal policy.

3.2 Use Case: Topology Augmentation

In the topology augmentation problem the data center consists of a fixed hierarchical topology with electrical packet switches and an optical switch, which connects all the top-of-rack switches. While the optical switch is physically connected to all ToR switches, unfortunately, the optical switch can only support a limited number of active links. Given this limitation, the rules for the topology problem are defined as: (i) The model must select K links to activate at a given step during the simulation. (ii) The model receives a reward based on the link utilization and the flow duration. (iii) The model collects the reward on a per-link (optical link) basis after x seconds of simulation. (iv) All flows are routed using equal-cost multi-path routing (ECMP).

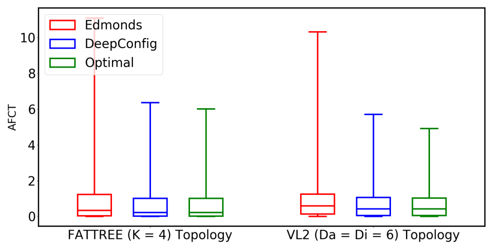
State Space: The agent specific state space is the network topology, which is represented by a sparse matrix where entries within the cells correspond to active links within the network.

Action Space: The agent interacts with the environment by adding links between ToR switches. The *action space* for the model, therefore, corresponds to the different possible link combinations and is represented as an n dimensional vector. The values within the vector correspond to a probability distribution, where w_i is equal to the probability of link i being the optimal pick for the given input state s. The model selects the highest K values from this distribution as the links that should be added to the network topology.

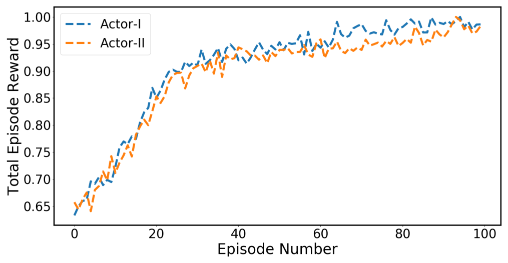
Reward: The goal of the model can be summarized as: (1) Maximize link utilization and (ii) Minimize the average flow-completion time. Hence, we formulate our reward function as:

$$R(i) = \sum_{f \in F} b_f \tag{3.1}$$

where F represents all active and completed flows during the previous iteration step, i represents the specific optical link, b_f represents the number of bytes transferred during the step time. The reward is collected separately for each link and the overall reward is represented as an n dimensional vector. The purpose of this reward function is to reward for high link utilization which directly would minimize the average flow-completion time.



(a) Average FCT



(b) This figure shows that the model is able to maximize the rewards over the training period.

Figure 2: DeepConf Performance

Model Configuration: As shown in Figure 1b, each input space has a set of CNN blocks which is fed into shared fully-connected layer, value network and policy-network. Table 1, presents the configuration for the inputs space and the various layers in the model where D is the number of ToRs in the data center and O is the number of potential optical links from which K are selected.

4 Evaluation

Experiment Setup We evaluate DeepConf on a trace driven flow-level simulator using a large scale map-reduce traces from Facebook [2].We evaluate two state-of-the-art clos-style data center topologies: K=4 Fat-tree [4] and VL2 [12]. In our analysis, we focus on flow completion time (FCT) a metric which captures the duration between the first and last packet of a flow. We augment both topologies by adding an optical switch with five links. We compare DeepConf against: **Optimal**, the optimal solution derived from a brute-force program with perfect knowledge of future application demands [1] and **Edmonds** a graph-based heuristic used in several existing works [9, 14, 23].

Model Learning Figure 2b presents the performance of each independent agent and we observe that each agent independently maximizes the total reward for each episode. Around episode 40-60, the performance of both agents begins to plateau as the policy becomes fine-tuned towards maximizing the reward. The training results demonstrate that the RL agent learns to optimize its policy decision to increase the total reward received across each episode.

DeepConf Performance Figure 2a shows that DeepConf is able to learn a solution that's close to the optimal [23, 9] across representative data center topologies while outperforming Edmonds, a greedy heuristic used by many topology augmentation papers [23, 9] in terms of FCT.

5 Conclusion

In this paper, we take the first steps towards moving away from heuristic-based approaches by designing a reinforcement learning based framework, called DeepConf, for learning and implementing a range of data center networking techniques. We believe that DeepConf represents a promising step towards a datacenter framework that eschews human generated domain-specific heuristics in favor of automated and general algorithms generated by a machine learning framework. As part of future work, we are planning to extend DeepConf to tackle the routing problem, to perform a broader evaluation of DeepConf across different topologies, and to explore the use of interpretable techniques to explain the decisions made by DeepConf's models.

[1]This optimal solution can not be solved with larger topologies [6]

References

[1] DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. `https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre- cooling-bill-40/`.

[2] Statistical Workload Injector for MapReduce . `https://github.com/SWIMProjectUCB/SWIM/wiki`.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of ACM SIGCOMM 2008*.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of ACM SIGCOMM 2008*.

[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of USENIX NSDI 2010*.

[6] T. Benson, A. An, A. Akella, and M. Zhang. Microte: The case for fine-grained traffic engineering in data centers. In *Proceedings of ACM CoNEXT 2011*.

[7] J. A. Boyan, M. L. Littman, et al. Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in neural information processing systems*, pages 671–671, 1994.

[8] A. Das, C. Lumezanu, Y. Zhang, V. K. Singh, G. Jiang, and C. Yu. Transparent and flexible network management for big data processing in the cloud. In *Proceedings of ACM HotCloud 2013*.

[9] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of ACM SIGCOMM 2010*.

[10] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian. Micro load balancing in data centers with drill. In *Proceedings of ACM HotNets 2015*.

[11] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, Dec. 2008.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of ACM SIGCOMM 2009*.

[13] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall. Augmenting data center networks with multi-gigabit wireless links. In *Proceedings of ACM SIGCOMM 2011*.

[14] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proceedings of ACM SIGCOMM 2014*.

[15] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *Proceedings of USENIX NSDI 2010*.

[16] S. A. Jyothi, C. Curino, I. Menache, S. M. Narayanamurthy, A. Tumanov, J. Yaniv, R. Mavlyutov, I. Goiri, S. Krishnan, J. Kulkarni, and S. Rao. Morpheus: Towards automated slos for enterprise clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 117–134, GA, 2016. USENIX Association.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[18] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2016.

[19] 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.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[21] N. Usunier, G. Synnaeve, Z. Lin, and S. Chintala. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*, 2016.

[22] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 185–191, New York, NY, USA, 2017. ACM.

[23] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. In *Proceedings of ACM SIGCOMM 2010*.

[24] D. Zhuo, M. M. Ghobadi, R. Mahajan, K.-T. Forster, A. Krishnamurthy, and T. Anderson. Understanding and mitigating packet corruption in data center networks. page 14. ACM SIGCOMM, August 2017.