Learning-based Network Performance Estimators: The Next Frontier for Network Simulation

Kai Shen , Baochun Li

***Abstract*—Over the past few decades, a tremendous amount of research attention has been received to derive the network performance estimation problem. In its context, network performance estimators can provide an early-stage prediction before emulation and real-world deployment, which is essential for network design and optimization. The design philosophy of network performance estimators is to design accurate estimators with scalability and generality. However, conventional rule-based network simulators are not able to satisfy all these demands simultaneously. To achieve these objectives, it has become an inevitable and appealing trend to empower network performance estimators with machine learning, especially with deep learning techniques. In this article, we present a cursory glimpse of existing results over the past five years in learning-based network performance estimators, with a particular focus on understanding the current challenges, the basic ideas and issues of state-of-the-art solutions, and essentially, the open challenges and future directions in research attention.**

I. Introduction

Network simulation is one of the most fundamental and challenging problems in computer networking research. In the context of network simulation, network performance estimators act as a critical component and have evolved over a few decades. It can provide an early-stage network performance estimation before emulation and real-world deployment, which is essential for future network architecture optimization, such as topology design and device parameter tuning. The primary design objective with network performance estimators is to be as *accurate* and *scalable* as possible. Conventional rule-based simulators fail to meet such an objective: they are accurate, but not necessarily scalable. Emerging deep learning techniques offer a promising alternative.

Over the past few decades, a tremendous amount of research attention has been received to derive the network simulation problem. The community has resorted to two different directions of research towards rule-based network simulation, which are illustrated in Table I. The first category is discrete event simulation (DES), which serves as the most classic type of network simulation. Typical DES-based simulators include ns-3 [1], OMNeT++ [2], and OPNET [3]. With packet-level granularity, DES explicitly simulates each packet and its associated events, enabling a comprehensive representation of all network components and providing high simulation accuracy. However, DES suffers from its scalability when we encounter large-scale network simulation tasks, like FatTree [4] topology for data center networks. Although the direction of Parallel and Distributed Discrete Event Simulation (PDDES) [5] has been carefully explored to alleviate the synchronization overhead associated with the communication traffic between different processes, there is still no speedup guarantee due to the nature of synchronization algorithms being applied and the architecture of the computing platform in use.

Kai Shen and Baochun Li are with the University of Toronto.

TABLE I
Characteristics of Rule-based Network Simulators.

Simulators	Accuracy	Granularity	Scalability
DES [1]–[3]	high	packet-level	low
Fluid models [6]	low	flow-level	high
Network calculus [7]	low	flow-level	high
Queueing-theoretic models [8]	high	packet-level	low

The second network performance estimation approach can be categorized as continuous simulation. As the name implies, continuous simulation focuses on interpreting the abstraction of traffic flows in the target network. Many attempts have been made towards continuous simulation, and they can be divided into three branches.

The first branch of continuous simulation is stochastic fluid models [6], an abstraction that simplify the representation of data flows by treating it as a continuous fluid. These fluid models are efficient as a flow-level solution which offer computational efficiency, ease of analysis, and scalability when compared to packet-level simulations. However, these benefits are obtained at the expense of accuracy. Secondly, Network calculus [7] is a theoretical flow-level analytic method that offers a mathematical framework to reveal worst-case bounds on network performance, such as delay, jitter, and loss. Nevertheless, network calculus requires precise bounds of arrival traffic patterns to derive performance metrics, which is challenging in increasingly complex network environments. Lastly, the most advanced continuous simulation approach is queueing-theoretic models [8], which allows for accurate simulation by characterizing system behavior using a queueing model comprising packet arrival modeling and scheduling server modeling. With respect to packet-level queueing theories, simulators can easily imitate the behavior of the whole network, including packet arrivals and scheduling disciplines. Hence, it can accurately predict the delay and loss for each packet. Such an extremely effective means nevertheless encountered the scalability issue. That is, its computation overhead is exponentially dependent on both the number of queues and their buffer sizes, rendering it impractical for large-scale network performance estimations.

With the rapid development of deep learning, there are recent research interests towards moving away from the conventional wisdom of using rule-based predictions, and shifting to the use of deep learning techniques for network performance

estimation. The essence of applying deep learning algorithms is to replace the computationally expensive parts of prior work with deep neural networks (DNN), so that such learning-based network performance estimators can achieve satisfactory accuracy with high scalability. It is worth emphasizing that the ultimate design objective is to accurately estimate network performance with scalability and generality. Over the past five years, it has become a trend in academia to propose high-quality network performance estimators by leveraging deep learning algorithms.

In this article, we present a concise survey of recent prominent research on the growing popularity of proposing network performance estimators based on deep learning techniques. We highlight several key challenges, share three state-of-the-art learning-based network performance estimators followed by their corresponding issues. Furthermore, we provide insights toward potential future directions in this field.

II. Examining the Challenges of Learning-based Network Performance Estimators

To date, existing research on discrete event simulation and continuous simulation has demonstrated notable performance in either simulation accuracy or scalability, not yet both. Inevitably, their performances are still under satisfaction in the face of large-scale networks because of their inherent drawbacks. Getting away from previously commonly adopted rule-based network simulators, we notice a trend in encapsulating deep learning techniques to construct the next frontier of network performance estimators. Taking full advantage of deep neural networks, next-generation network performance estimators can be both accurate and scalable. However, considering the high performance estimation demand and the widely diverging quality of different networks, there are still several critical challenges that should be seriously considered and carefully addressed.

Accuracy: The accuracy of learning-based network performance estimators is the most straightforward and explicit evaluating metric. While deep learning techniques showcase the potential to produce accurate observations, we still need to guarantee that the learning-based estimators are sufficiently trained and can truly be comparable to those of packet-level simulators, i.e., discrete event simulators. Additionally, to maximize accuracy, one possible approach is to increase the interpretability of the overall network structure. That is, attempting to capture the connections between the final evaluation metrics and the characteristics of traffic flows, network topologies, and device configurations. Higher interpretability will lead to better employment of deep learning techniques, which can also serve as an essential challenge for accuracy improvement.

Scalability: Scalability is another essential challenge for network performance estimators. In order to design scalable estimators to be employed in large-scale networks, we seek to focus on lowering computational and communicational burdens, so as to increase training efficiency. As accurate DES and queueing-theoretic models are computationally-complex to scale beyond one or a few devices, deep learning techniques exhibit considerable application potential due to their constant time complexity during model inference. In that way, one of the promising design philosophies is parallelism. With current distributed and parallel deep learning frameworks, the inference of learning-based estimators can be accelerated easily. However, only parallelism itself is not enough, and how to combine other training approaches reasonably on a parallel basis to improve efficiency is also a decisive challenge.

Generality: More importantly, the next frontier of network performance estimators should be generalizable. Ideally, such learning-based estimators should be applicable to arbitrary network designs, encompassing network topologies, traffic patterns, and network device configurations. Additionally, they should be able to predict a variety of metrics, such as delay, jitter, and loss. It is worth noting that retraining the entire estimator model when facing even minor network changes is not economical. Thus, developing reliable and easily adjustable learning-based estimators with generality is a substantial challenge.

III. Basic Ideas and Issues of Advanced Solutions

In this section, we walk through three representative frameworks for DNN-based network performance estimators from the research community. With our investigation of how deep neural networks are leveraged through the lens of recent literature, we have seen that they created a trend to replace a specific portion of network architectures with advanced deep learning models for the next frontier of network performance estimation. Again, the ultimate design objectives are to accurately estimate network performance with scalability and generality.

A. RouteNet

In network performance estimation tasks, traditional simulators struggle to provide functional networks models for accurate predictions. To bridge this gap, it is conceivable that deep learning algorithms can be used to understand the complex relationships among topology, routing, and input traffic, thereby producing accurate estimates of key performance indicators (KPIs) like delay distribution and loss.

As one of the first attempts toward this direction, Rusek et al. [9] designed a new framework, named *RouteNet*, to predict key performance indicators of the whole network by employing Graph Neural Networks (GNN) [10]. The high-level idea is to build the network as a graph, where each edge represents a link between network devices (e.g., routers, switches). In particular, GNN is leveraged to process the graph so as to capture the complex relationships in each path given the network topology and routing configurations. According to relational reasoning and combinatorial generalization over graph-based information structure, *RouteNet* can estimate the performance of arbitrary topologies, routing schemes and variable traffic intensity.

Figure 1 demonstrates that *RouteNet* accommodates variable-size network topologies, arbitrary source-destination routing schemes, and traffic matrix as input while predicting end-to-end key performance indicators as output. In details,

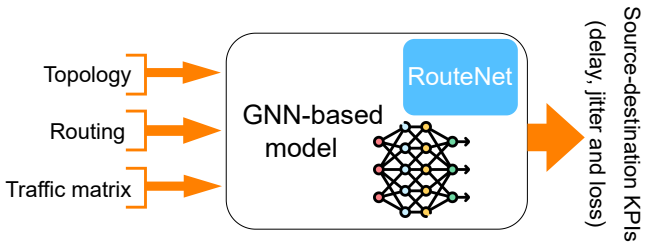

Fig. 1. The core architecture of RouteNet is a GNN-based model. It takes topology, routing, and traffic matrix as input, and outputs source-destination key performance indicators.

RouteNet treats the entire network between the source and the destination as a black-box, and employs GNN models to construct the message-passing architecture specifically tailored to produce accurate performance estimates. The model produces two types of output: packet-level information, such as end-to-end delay representing packet sojourn time from source to destination, and link-level information, which includes statistics for each link or path, like packet drop rate over a period of time for each pair of devices. Note that the main assumption behind *RouteNet* is that the information for all links, as well as paths that are constructed by links, can be encoded in learnable vectors of real numbers. Based on this assumption, *RouteNet* is built upon the following two principles:

- The state of a path depends on the state of all the links that lie on the path.
- The state of a link depends on the state of all the paths that traverse the link.

As one of the early-stage explorers to build network performance estimators upon deep neural networks, there are a number of clear advantages brought forth by *RouteNet*. The most salient advantage is its prediction capability over arbitrary topologies, routing schemes and variable traffic intensity. Secondly, benefiting from its network-scale modeling, which encompasses the entire network in a single model, *RouteNet* is scalable by performing with low computational cost in acceptable time budgets. Hence, *RouteNet* can be used for Quality of Service (QoS)-aware routing optimization tasks by evaluating the resulting performance after testing new routing modifications. In addition, it can also be useful to explore the optimal network device upgrading problem.

With the design philosophies in mind, we argue that *RouteNet* has the potential to perform better in many aspects. It is intuitive to point out that the scope of the network-scale GNN-based model can be narrowed down to achieve more interpretability of networks, and thus achieve better estimation performance. For the accuracy challenge, *RouteNet* only performs well on small-scale networks, and it is far from satisfaction when facing more complicated network topologies. Although *RouteNet* supports a variety of QoS-aware performance metrics that include delay, jitter, and packet drops, it is impossible to provide more flexibility toward extra accurate performance indicators. For instance, the quantile-based end-to-end measure, which is not sub-additive. Moreover, *RouteNet* is devoid of higher generality due to its inherent drawback of using specially trained model for each network. Even though the computational overhead is manageable, the customized source-destination settings restrict the possibilities to reuse pre-trained models, posing a challenge for rapid model deployment on large-scale networks.

B. MimicNet

RouteNet has shown that there is a crucial interplay between deep learning techniques and network performance estimators. Continuing their work on modeling the whole network by deep neural networks, Zhang et al. [11] proposed *MimicNet* by combining DES and deep learning techniques. Similar to *RouteNet*, *MimicNet* is inspired by providing users with an abstraction of the simulation for a portion of the network while leveraging the advances in deep learning techniques. A significant difference in the motivation of *MimicNet*, however, is the severe scalability issue of modern networks. Throughout its observations, modern networks, especially data center networks, connect up to hundreds of thousands of machines, which should be capable of processing hundreds of billions of packets per second in aggregation. As a consequence, *MimicNet* targets on scale-out network architectures, in particular, FatTree topology. In essence, FatTree topology is a layered network architecture, consisting of core, aggregation, and edge layers with multiple parallel paths between nodes. In addition, a cluster refers to a pod of switches in the edge and aggregation layers, and in a k-pod FatTree, each cluster contains k switches with k ports per switch.

As proposed to address the scalability issue in scale-out data center networks, *MimicNet* is believed to have two outstanding design shining points. Firstly, *MimicNet* sheds the first light on designing accurate and scalable network performance estimators by reducing the scope of the DNN from network scale to cluster scale. By constructing and composing models at the granularity of individual data center clusters, which are named as mimics, *MimicNet* employs DNNs to model their performance. This approach allows it to perform packet-level accurate simulations for each cluster within the data center network. In more detail, as a cluster-level estimator, *MimicNet* removes the observable effects of internal traffic and bake its effects into the cluster-level predictions. Therefore, *MimicNet* uses mimics to resemble regular clusters and predicts how the network of the cluster will affect packets on the basis of external traffics.

The second significant design philosophy of *MimicNet* is to combine DES and continuous simulation, as well as deep learning techniques. It is worth noting that the goal of *MimicNet* is not to replicate the effects of large-scale network simulation, but to generate results that are able to exhibit their characteristics. Hence, it is natural to conceive a simple but effective design of network estimators, in which the system, is coupled by two key models responsible for packet-level simulation and network behaviors learning, respectively. DES finishes the packet-level accurate simulation of clusters, and its output includes detailed queueing and transport layer dynamics. Afterwards, deep-learning-based internal models will be leveraged to train mimics that learn both non-observable

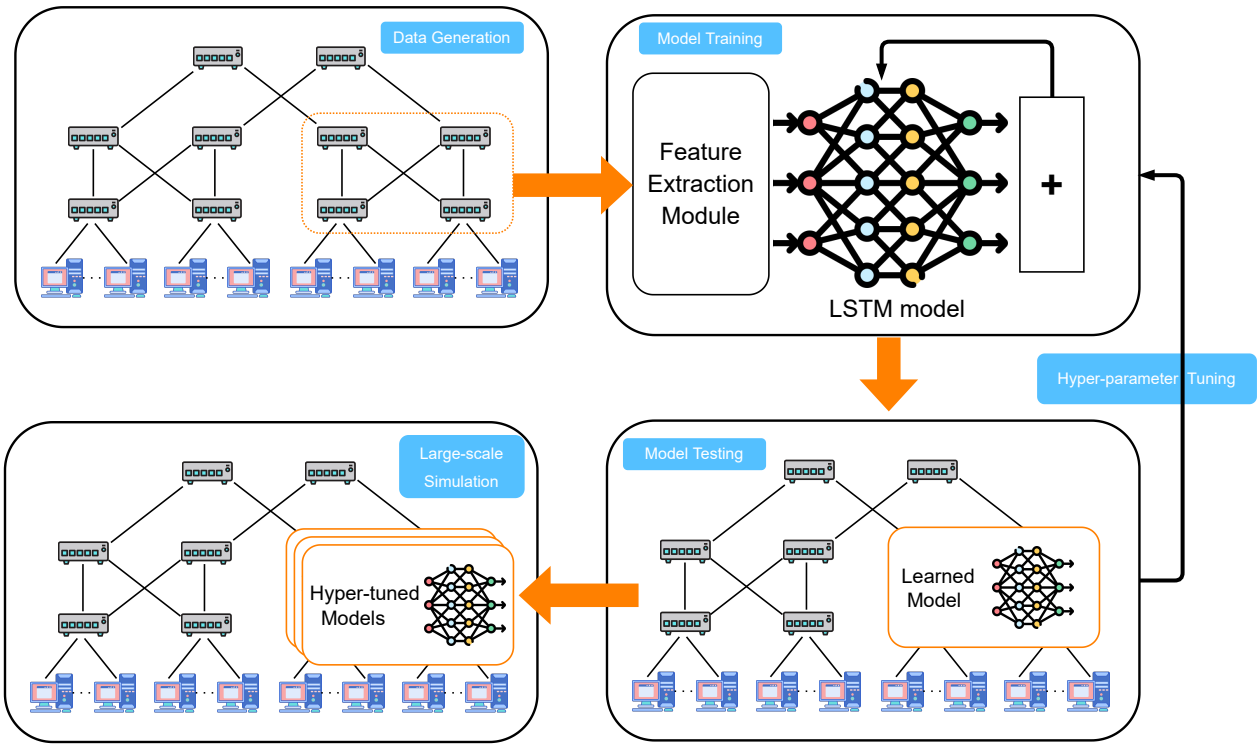

Fig. 2. The workflow of MimicNet includes 5 steps: a) data generation through small-scale observations; b) model training; c) model testing; d) hyper-parameter tuning to produce tuned LSTM models for use in mimics, which speed up large-scale simulations; e) large-scale simulation by replacing the majority of the network.

internal and cross-cluster behaviors based on the collected data.

Figure 2 illustrates the end-to-end, fully automated workflow of *MimicNet*. There are five steps in the full workflow of *MimicNet*: data generation, model training, model testing, hyper-parameter tuning, and large-scale simulation. The usage of *MinicNet* begins with a small-scale data generation. In such a small subset of the full simulation, two clusters will communicate with each other, and the generated full-fidelity, small-scale simulation results will be utilized for later model training and testing stages. Next, *MimicNet* adopts Long Short Term Memory (LSTM) [12] models as the core, which will be trained and tested recursively. Augmenting this training phase is a configurable hyper-parameter tuning stage in which *MimicNet* explores various options for modeling. With respect to the hyper-parameter tuning stage, *MimicNet* aims to maximize the accuracy of end-to-end metrics like throughput and Flow-Completion Time (FCT), as well as increasing the generality to larger configurations and different traffic matrices. The final step involves replacing clusters in large-scale networks with tuned mimics to accelerate simulation and increase prediction accuracy. Also, it is worth noting that a salient feature of *MimicNet* is that the first four steps are all done at small scale, which serve as the main reason why *MimicNet* is able to outperform prior works in rapid training and deployment.

Performance-wise, *MimicNet* achieves high accuracy on all metrics, including FCT, throughput and Round-Trip Time (RTT). Across the entire range, the Cumulative Distribution Function (CDF) of *MimicNet* adhere closely to the ground truth (i.e., the full-fidelity, packet-level simulation), and behaves much better than other continuous simulators. It is reasonable to achieve such high accuracy as it utilizes DES to accurately simulate a small subset of the scale-out network, and thus can obtain and feed the LSTM model with detailed information such as queueing and protocol interactions. With respect to the scalability, *MimicNet* performs surprisingly good that it can provide consistent speedups up to 675× for the largest network that full-fidelity DES simulators could handle, which has 128 clusters in a FatTree topology. Above that size, full-fidelity DES could not finish within three months, while *MimicNet* can finish in under an hour. From this point of view, the scalability of *MimicNet* is incredibly remarkable, benefiting from its substantial design philosophy that models each cluster as a whole.

Although *MimicNet* achieves orders of magnitude reduction in simulation completion time of modern scale-out data center networks, its generality is still a severe concern. Along this direction, we argue that *MimicNet* only works for FatTree networks. Hence, we question its generality, and believe that it is not able to be agile in adapting to more complex and challenging network topologies.

C. DeepQueueNet

Continuing the prevailing work of *MimicNet*, Yang et al. [13] set their sights on further improving the generality of network performance estimators and proposed DeepQueueNet.

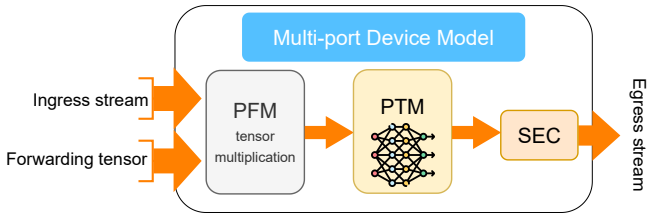

Fig. 3. There are three parts in multi-port device model of DeepQueueNet: packet-level forwarding model, packet-level traffic-management model, and statistical error correction model. The overall device model takes ingress stream and forwarding tensor as input, and outputs the egress stream.

There are three insights of their estimator design. First, based on the success of the predecessor that achieves high accuracy and scalability by narrowing down the scope of DNN-based estimators from the network level to cluster level, *DeepQueueNet* is motivated to further narrow down the scope of DNN-based estimators to device level. Second, *DeepQueueNet* seeks to address the unsolved generality issue, making it more cost-effective to adapt to changes in network settings. This ensures that network performance estimators will not be confined to a specific topology. The last insight lies in providing packet-level statistics to reveal concrete performances about specific devices, flows, or even packets. In other words, *DeepQueueNet* aims to promote performance visibility to the packet level.

With the goal of achieving accurate, scalable, and general deep-learning-based network estimators with packet-level visibility, *DeepQueueNet*, combines prior networking knowledge and advanced simulation techniques. More specifically, it starts with solid queueing-theoretic modeling of networks, and utilizes deep neural networks to model the mathematically-intractable or computationally-expensive parts.

To run the simulations with the underlying deep learning framework, there are basically three steps in the workflow. The first step is simulation preparation and device modeling. Similar to DES, we need to prepare simulation settings to feed *DeepQueueNet*, which include network topologies, device configurations, and traffic generators. Meanwhile, as a device-scale network performance estimator, *DeepQueueNet* trains and maintains a device library. In this library, deep learning algorithms are applied to predict the input-output relationships of various network devices, such as routers and switches. As such, users can easily retrieve device models from the pre-trained device library based on their specified device configurations. Consequently, *DeepQueueNet* can own the packet-level visibility from the predicted results of each device while maintaining the generality across network devices and traffic patterns. Afterwards, the second step is to compose the network. Given the network topology, *DeepQueueNet* will map it to a corresponding neural network architecture. Hence, it surpasses *MimicNet* in terms of the generality for network topologies because of its flexibility in device setting within the network. Lastly, *DeepQueueNet* will run parallel inference. That is, decomposes the network for parallel inferences. It proposed a core execution algorithm to guarantee the correctness of the framework running, and greatly increased the scalability to network sizes as it processes packets in batches.

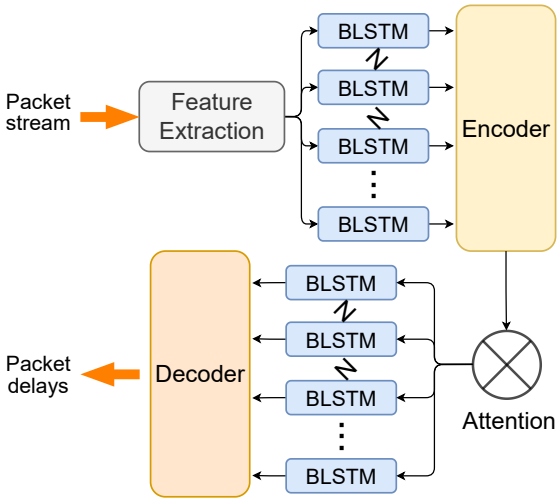

Fig. 4. The DNN architecture of the packet-level traffic management model.

One of the most prominent design of *DeepQueueNet*, we argue, is about the device modeling. There are three types of devices considered in this framework: link device, switch device and router device. For the simplest link device with only one input and one output port, *DeepQueueNet* regards it as an operator that adds a latency to all packets in the ingress time series, where the latency calculation is based on the length of each packet and the characteristics of the link (i.e., the length, the bandwidth, and the propagation speed of the link). Generally, multi-port network devices take the packet stream of each ingress port as the first input, obtain a forwarding tensor as the second input, and output packet streams of all egress ports to represent the device-level simulation. Notably, the forwarding tensor acts as an indicator of the forwarding path from the ingress port to the the egress port of all packets.

As illustrated in Figure 3, multi-port network device models have two sub-models: packet-level forwarding model (PFM) and packet-level traffic-management model (PTM). The packet-level forwarding model specifies the forwarding behavior of the device, that is, to describe the device behavior explicitly using tensor multiplication of the given forwarding tensor and the ingress stream information. The advantage of adopting such PFMs is to enable high scalability by forwarding in batches, while previous studies could only process each packet sequentially.

With respect to PTM, its model will predict how much delay is experienced for each packet. Given the pre-processed packet vector, it will perform inference from pre-trained DNN models and add delays to each packet in batches. As a sequence-to-sequence processing task, *DeepQueueNet* chose the Transformer architecture [14] with attention mechanism to train the PTM. Figure 4 reveals the detailed DNN architecture of the PTM. According to the generated packet-level training data from DES simulators, training PTM follows the approach of regression in forecasting the delay a packet experiences in a device. The sojourn time of each packet serves as the response variable, and the features extracted from the feature extraction model are regarded as the predictors. In implementation details, *DeepQueueNet* selects a 2-layer Bidirectional Long Short

Term Memory (BLSTM) [15] cell for the Decoder-Encoder mechanisms, and three parallel heads are jointly attended to the information from different representation subspaces at different positions. In addition, *DeepQueueNet* also designed a statistical error correction (SEC) model as the post-PTM part, where the effect of the accumulated errors of the predicted sojourn time will not be propagated along the packet transmission path.

As the state-of-the-art network performance estimator, *DeepQueueNet* is superior to all previous simulators in many aspects. Firstly, it provides accurate delay distributions in an end-to-end fashion, which is close to the ground-truth in many modern networks. Secondly, *DeepQueueNet* achieves packet-level visibility, which help to realize expressive device models, process packets in batches, and predict the input-output relationship. Besides, *DeepQueueNet* has better generality than all previous deep-learning-based estimators. Its generality performs well for different traffic generation models, topologies, and traffic management mechanisms. Last but not least, *DeepQueueNet* is scalable because of its ability of parallel inferences and its near-linear speedup with the number of additional GPUs.

However, there are still some notable disadvantages that cannot be overlooked. The most obvious limitation of *DeepQueueNet* is that it cannot deal with stateful behaviors of network devices. This limitation is because the network-layer design will ignore the complex interactions among the high layers, such as the underlying transport protocols, the network layer traffic management mechanisms, as well as any possible interaction between devices. Moreover, the *DeepQueueNet* puts aside the prediction of packet drops, limiting its applicability to networks with only well-behaved performances (i.e., no packet drop), thus reducing its practical value in realistic environments.

D. Recap

In summary, *RouteNet* takes the initial step in utilizing DNNs for network simulation. Expanding on this idea and targeting data-center networks, *MimicNet* reduces the network portion that DNNs mimic from the entire network to the clusters, significantly enhancing scalability. Finally, the most advanced *DeepQueueNet* models network devices, substantially improving accuracy and generality while maintaining satisfactory scalability. Among these three work, *MimicNet* exhibits the highest scalability due to its original focus on scale-out networks, whereas *DeepQueueNet* possesses the highest accuracy and generality.

IV. Open Challenges and Future Directions

In retrospect, the meteoric rise of research interests in deep-learning-based network performance estimators was largely fueled by the rapid maturation of deep learning technologies, as well as by the need of accurate network simulation, most notably in large-scale modern networks. The essence of deep-learning-based estimators is to take full advantage of both solid queueing theories and advanced deep learning techniques, realizing potentially accurate, scalable ones with generality on network simulation tasks. However, there are many open technical challenges in the next frontier of estimators to be addressed before fully realizing its benefits.

Better generality. While recent estimators have demonstrated good performance in their experiments, there is still room for improvement in terms of generality. For instance, next-generation learning-based estimators should focus on generalizing more complex network devices. Even though *DeepQueueNet* is generalizable to arbitrary topologies, traffic patterns, and network devices, we argue, it still needs to maintain a relatively large device library for different switch models. Consequently, it lacks the generality to adapt to new device models that have not been previously trained with the corresponding parameters. Worse, these estimators have not been well and fully evaluated under realistic modern network scenarios.

Better interpretability. Another visible roadblock to realistic performance improvement of deep-learning-based estimators is the interpretability of their design. Reflecting on the development of network performance estimators, increased interpretability was achieved by narrowing of the application scope of DNNs, which substantially improved the accuracy of the prediction. Hence, exploring ways to better interpret the network through a deeper understanding of networking theories is a worthwhile question to address, as it may enhance the rationality of estimators and consequently improve prediction performance.

In light of the aforementioned open challenges, we discuss the following potential research directions for future studies.

Accommodating more devices. As mentioned above, higher interpretability can help increase accuracy as it leverages DNNs to imitate more reasonable modules. Thus, it is one of the promising directions to use DNNs to perform more black-box modeling of traffic management mechanisms on the device scale. In fact, it is a meaningful research direction to model more complicated packet-processing network devices. For example, modeling some network devices that can split or combine packets.

Narrowing down the scope of DNNs. Another promising approach is to further narrow down the scope of DNNs by interpreting the components of each network device and using DNNs to replace only the computationally complex parts. For instance, improved generality could be achieved by utilizing DNNs to represent only the ports in a switch. In that case, network devices could be composed initially, followed by connecting the network using the composed devices and given topologies. As a result, maintaining a large device library would be unnecessary, potentially leading to increased generality.

Applications for network optimization. In addition to building estimators with higher scalability and better generality, we can also focus on applying them to network optimization tasks. To some extent, it is an inevitable and appealing trend to leverage the predictions of network performance estimators to apply modifications in a network configuration. In particular, we point out that potential optimization tasks include but are not limited to the following two problems. Firstly, how to optimize the QoS-aware routing based on the

simulation results? Secondly, how to optimize the network by adding new links in the topology or upgrading network devices.

V. Acknowledgments

The research was supported in part by a RGC RIF grant under the contract R6021-20, and RGC GRF grants under the contracts 16209120, 16200221 and 16207922.

References

[1] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," *Modeling and tools for network simulation*, vol. 14, 2010, pp. 15-34.
[2] A. Varga, "A Practical Introduction to the OMNeT++ Simulation Framework," *Recent advances in network simulation*, 2019, pp. 3-51.
[3] Z. Lu and H. Yang, "Unlocking the Power of OPNET Modeler," *Cambridge University Press*, 2012.
[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *Proc. ACM SIGCOMM*, 2008, pp. 63-74.
[5] S. Jafer, Q. Liu, and G. Wainer, "Synchronization Methods in Parallel and Distributed Discrete-event Simulation," *Simulation Modelling Practice and Theory*, vol. 30, 2013, pp. 54-73.
[6] V. Misra, W. Gong, and D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," *Proc. ACM SIGCOMM*, 2000, pp. 151-160.
[7] F. Ciucu and J. Schmitt, "Perspectives on Network Calculus – No Free Lunch, but Still Good Value," *Proc. ACM SIGCOMM*, 2012, pp. 311-322.
[8] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. the IEEE*, vol. 83, no. 10, 1995, pp. 1374-1396.
[9] K. Rusek, J. Suarez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet: Leveraging Graph Neural Networks for network modeling and optimization in SDN," *IEEE JSAC*, vol. 38, no. 10, 2020, pp. 2260-2270.
[10] F. Scarselli, M. Gori, A. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE TNN*, vol. 20, no. 1, 2008, pp. 61-80.
[11] Q. Zhang, K. K. Ng, C. Kazer, S. Yan, J. Sedoc, and V. Liu, "MimicNet: Fast Performance Estimates for Data Center Networks with Machine Learning," *Proc. ACM SIGCOMM*, 2021, pp. 287-304.
[12] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM", *Neural Computation*, vol. 12, no. 10, 2000, pp. 2451-2471.
[13] Q. Yang, X. Peng, L. Chen, L. Liu, J. Zhang, H. Xu, B. Li, and G. Zhang, "DeepQueueNet: Towards Scalable and Generalized Network Performance Estimation with Packet-level Visibility," *Proc. ACM SIGCOMM*, 2022, pp. 441-457.
[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[15] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," *arXiv preprint arXiv:1508.01991*, 2015.

Biographies

Kai Shen is currently pursing his M.A.Sc. degree at the Department of Electrical and Computer Engineering, University of Toronto. He received his B.Eng. degree in Computer Science and Engineering from The Chinese University of Hong Kong, Shenzhen. His research interests include networking, deep learning and federated learning.

Baochun Li received his B.Eng. degree from Tsinghua University and his M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign. He is a professor in the Department of Electrical and Computer Engineering, University of Toronto. He is a Fellow of IEEE and Fellow of the Canadian Academy of Engineering.