SL-Hyper-FleX: A cognitive and flexible-bandwidth optical datacom network by self-supervised learning [Invited]

To be published in Journal of Optical Communications and Networking:

Title: SL-Hyper-FleX: A Cognitive and Flexible-bandwidth Optical Datacom Networks by Self-supervised Learning

Authors: Che-Yu Liu,Xiaoliang Chen,Zhaohui Li,S. J. Ben Yoo,Roberto Proietti

Accepted: 16 November 21

Posted 17 November 21

DOI: https://doi.org/10.1364/JOCN.439801

SL-Hyper-FleX: A Cognitive and Flexible-Bandwidth Optical Datacom Networks by Self-supervised Learning [Invited]

CHE-YU LIU[1], XIAOLIANG CHEN[2], ZHAOHUI LI[2], ROBERTO PROIETTI[3,*], AND S. J. BEN YOO[3,*]

[1] Department of Computer Science, University of California, Davis, Davis, CA 95616, USA
[2] Sun Yat-Sen University
[3] Department of Electrical and Computer Engineering, University of California, Davis, Davis, CA 95616, USA
[*] Corresponding author: rproietti@ucdavis.edu, sbyoo@ucdavis.edu

Compiled October 29, 2021

This paper presents a cognitive flexible-bandwidth optical interconnect architecture for datacom networks. The proposed architecture leverages silicon photonic reconfigurable all-to-all switch fabrics interconnecting top-of-rack switches arranged in a Hyper-X-like topology with a cognitive control plane for optical reconfiguration by self-supervised learning. The proposed approach makes use of a clustering algorithm to learn the traffic patterns from historical traces. We developed a heuristic algorithm for optimizing the intra-pod connectivity graph for each identified traffic pattern. Further, to mitigate the scalability issue induced by frequent clustering operations, we parameterized the learned traffic patterns by a support vector machine classifier. The classifier is trained offline by self-labeled data to enable the classification of traffic matrices during online operations, thereby facilitating cognitive reconfiguration decision making. The simulation results show that compared with a static all-to-all interconnection, the proposed approach can improve the throughput by up to 1.62× while reducing the end-to-end packet latency and flow completion time by up to 3.84× and 20×, respectively.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

The rapid expansion and evolution of cloud-based services demand for data center (DC) and high-performance computing (HPC) architectures supporting high-bandwidth and low-latency communications among tens of thousands of servers. These systems are typically built with tree-based electronic packet switching (EPS) architectures (e.g., Fat Tree [1, 2] with point-to-point optical links for inter-rack communications. Fat Tree provides rich interconnectivity but suffers from high power consumption and end-to-end latency due to the cascaded switching stages and related optical to electrical to optical (O/E/O) conversions. Overall, sustaining the ever-growing demand for low-latency and large data movement (typical of scientific computing, emerging deep learning workloads, etc.) is a challenge for future cloud computing systems. As wavelength division multiplexing (WDM) transceivers are penetrating the datacom market (to sustain the ever-increasing switch port data rates - from 100G to 200G and beyond) and silicon-photonic (SiPh) technologies are becoming commercially viable [3, 4], there are opportunities to leverage photonics beyond point-to-point communication. In particular, it is possible to flatten the inter-rack network architecture with direct all-to-all wavelength routing and switching to enable superior energy efficiency and performance [5].

Researchers have reported several hybrid optical/electrical switching architectures. These solutions leverage different switching paradigms (from slow and centralized optical circuit switching of elephant flows [6] to fast and distributed synchronous [7] or asynchronous packet switching [8]). A different optical switching paradigm involves adapting the inter-rack connectivity and bandwidth (also called optical reconfiguration for bandwidth steering) to the traffic patterns. This reconfiguration allows removing link congestion due to hotspot traffic between specific rack pairs [9–11]. The switching operation does not happen on a packet or flow basis but only when the traffic characteristic changes significantly and over time scales larger than hundreds of microseconds or milliseconds ([12]).

The application-driven nature of the communication patterns between the computing nodes makes DC and HPC traffic exhibit high dynamicity and spatial nonuniformity [13, 14]. Therefore, effective reconfiguration schemes are desirable to fully exploit the benefits of optical interconnects and ensure efficient resource utilization. The authors in [9] proposed a hybrid optical/electrical switching-based reconfigurable network topology

leveraging heuristic algorithms for determining the wiring and bandwidth steering schemes for different traffic profiles. In [15], the authors leveraged a deep learning approach to learn the mapping between the traffic distribution and the optimal topology configuration. The authors in [16] exploited a deep learning module to predict the traffic in its virtual optical network and therefore can better allocate the resources for the network during reconfigurations. In [17], the authors applied deep reinforcement learning (DRL) to tackle the topology augmentation problem by training an agent to learn reconfiguration policies. The authors in [18] proposed an online scheme based on DRL to find the optimal network configuration. The work in [19] devised a DRL formulation according to dynamic DC traffic to meet the quality of service requirements of applications. Note that reconfiguration operations can cause traffic disruptions as they involve reconfiguring optical switches and routing tables in all related top-of-racks (ToRs). Hence, it is desirable to implement effective reconfiguration policies guiding when to perform the reconfiguration. The work in [20] applied DRL to learn autonomic reconfiguration policies from repeated trials and errors. However, the effectiveness of these approaches were only demonstrated under a small-scale system and can hardly scale. In [21, 22], the authors applied the threshold-based approaches to determine reconfiguration policies either by the number of packets waiting for cluster transmission or the requested QoS. However, to determine the reconfiguration point by the threshold-based approach is time-consuming and requires a large number of trials and errors. Our previous work in [12] proposed a cognitive reconfiguration policy relying on network performance (i.e., latency, packet loss rate) estimations by deep neural network (DNN) models. Nevertheless, training the DNN models requires collecting a large amount of performance data and labeling, introducing non-negligible operation overheads. Meanwhile, the approach still makes use of a fixed-threshold-based policy applied to the performance estimations.

In this paper, we extend the work in [23] and propose a self-supervised learning (SL) approach for cognitive reconfiguration decision making in a Hyper-X-type flexible-bandwidth optical interconnect architecture [5, 24]. We named this architecture SL-Hyper-FleX. The proposed approach can implement effective reconfiguration strategies by self-learning the characteristics of the traffic data. The goal is to learn and perform reconfiguration only when it is truly beneficial while avoiding excessive and costly reconfiguration operations that could be triggered when using a simple threshold-based policy. Section 2 introduces the enabling photonic technology and data plane architecture of SL-Hyper-FleX. Section 3 discusses the details of the control plane, reconfiguration algorithm, and strategies based on an SL methodology. Section 4 discusses the evaluation results. Finally, Section 5 concludes the paper summarizing the main results and discussing the future work.

2. SYSTEM ARCHITECTURE

This section describes the proposed flexible-bandwidth photonic DC architecture enabled by SL, named hereafter SL-Hyper-FleX. Fig. 1 shows Flex-LIONS, a SiPh bandwidth-reconfigurable all-to-all switching fabric at the core of SL-Hyper-FleX. Flex-LIONS contains an arrayed waveguide grating router (AWGR), microring resonator (MRR) add-drop filters, and a Mach-Zehnder (MZ) switching network [25]. For an $N\times N$ Flex-LIONS, an $N\times N$ AWGR provides all-to-all interconnection based on its wavelength-routing function. Next, b MRR drop filters at each input port of the AWGR are used to drop b WDM channels for bandwidth reconfiguration. Then, the dropped channels are spatially switched and added to the desired output port by an $N\times N$ broadband Beneš Mach-Zehnder switch (MZS) network and b MRR add-drop filters. In this way, the bandwidth between specific node pairs can be increased by a factor of b. Note that the maximum number of reconfiguration operations at any given time is equal to $N/2$. As demonstrated in [25], Flex-LIONS can leverage multiple free spectral ranges (FSRs) to maintain all-to-all interconnectivity before and after bandwidth reconfiguration. Before reconfiguration, all the WDM wavelengths in FSR0 and FSR1 of the AWGR are used for all-to-all communication (thanks to the cyclic feature of the AWGR). For bandwidth steering, only the wavelengths in FSR1 are switched by the MRR add-drop filters and the MZS network, while the wavelengths in FSR0 are untouched. In this case, all-to-all interconnectivity is always maintained through FSR0, guaranteeing that there will be no isolated nodes after a reconfiguration operation. Fig. 1(b) shows the wavelength allocation before reconfiguration (both FSRs maintain the all-to-all connection). Fig. 1(c) depicts an example of wavelength allocation after reconfiguration for FSR1. The colored add/drop rings represent the rings tuned to the colored wavelengths in Fig. 1(c) to achieve bandwidth steering.

As shown in Fig. 2(a)-(b), it is possible to use Flex-LIONS to interconnect a group of P ToR switches (referred hereafter as POD). By laying out the PODs in rows and columns and using multiple Flex-LIONSs in each dimension, we can build a Hyper-FleX network (a reconfigurable Hyper-X [5]) with fixed hierarchical all-to-all connectivity on FSR0 and reconfigurability on FSR1. If we assume state-of-the-art switch with $k = 128$ ports at 100 Gb/s, this 2D Hyper-X architecture can scale to $k^3/32 = 65,536$ servers while requiring optical switches with limited radix ($N = k/4 = 32$) and number of wavelengths ($N = k/4 = 32$). Furthermore, as discussed in [5], when using three dimensions, the architecture can scale to a larger number of nodes than a non-oversubscribed Fat Tree (for switch radix values higher than 128) while providing significant power savings (see [5] for more details).

There is one Flex-LIONS switch for each POD with a software-defined network (SDN) controller that interfaces with the Flex-LIONS switch and the group of ToR connected to it. There are P ToRs in a column POD and C ToRs in a row POD. As shown in Fig. 2(c), each ToR has P ports for intra-column POD communication and C ports for intra-row POD communication. The POD controller monitors the traffic demand for the ToR ports and leverages a combination of heuristic [12] and deep learning tools to determine the reconfiguration policies that best serve a specific traffic demand (see Section 3 for more details). After the POD controller calculates the new configuration, it will update the forwarding tables in the ToRs and drive the reconfiguration of the optical switch (in a similar way as presented in [26]). In this work, we are focusing on the self-supervised learning-aided reconfiguration operation within a single POD. More comprehensive designs performing reconfiguration across multiple PODs will be left as a future research task.

3. RECONFIGURATION DESIGN

A. Overall Principle

Fig. 3 shows the software-defined networking (SDN) control plane blocks with a reconfiguration module for a single POD. An SDN controller works with related ToRs and optical switch to collect the data plane states specified by the network monitoring

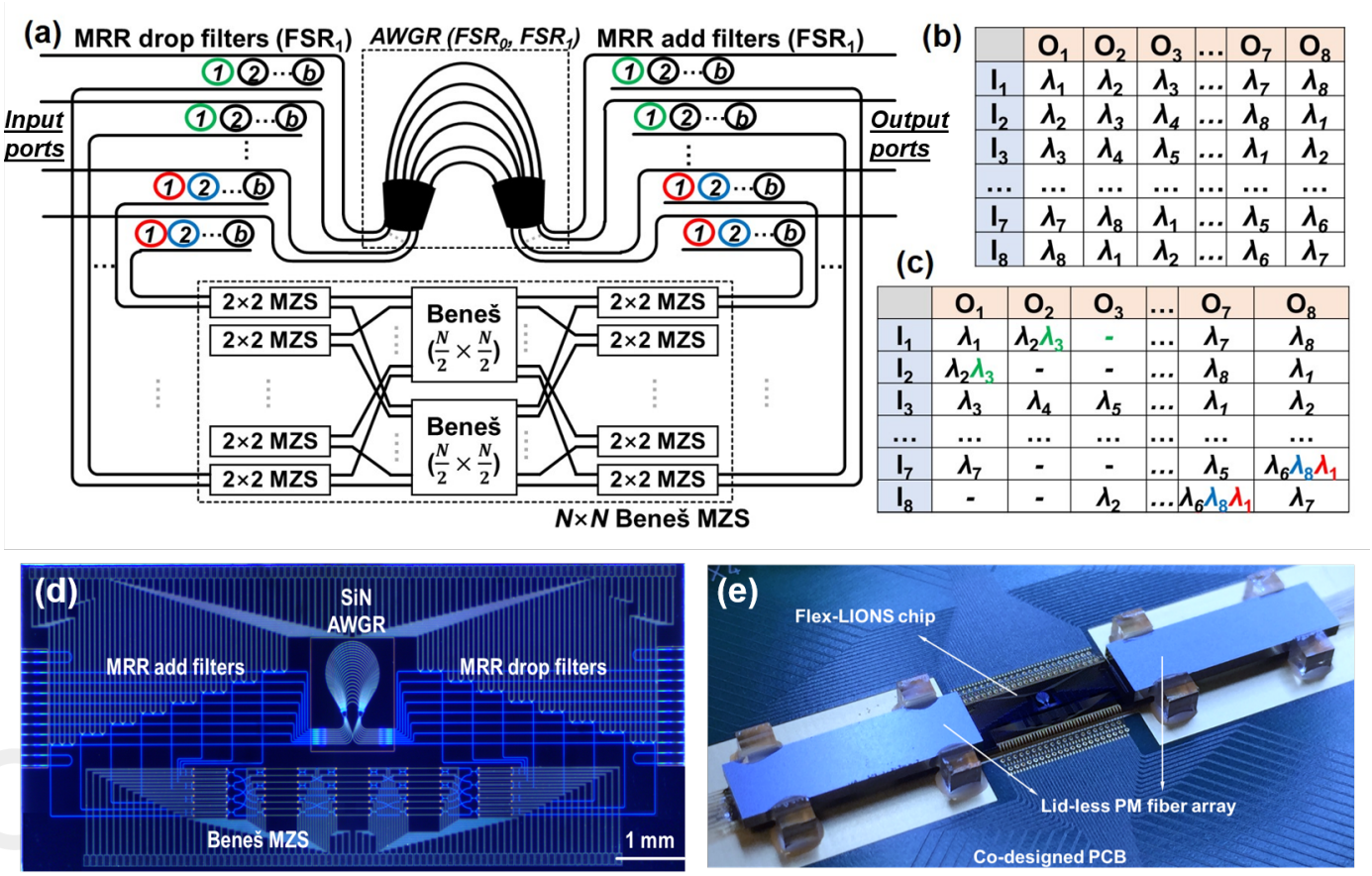

Fig. 1. (a) Block diagram of the Flex-LIONS device. Wavelength routing table (b) before and (c) after reconfiguration. The colored add/drop rings represent the rings tuned to the colored wavelengths in (c) to achieve bandwidth steering. (d) Microscope image of the fabricated 8 × 8 SiPh Flex-LIONS ($N = 8$, $b = 3$) chip. (e) Photograph of the integrated Flex-LIONS module with lid-less PM fiber arrays on a co-designed PCB (Courtesy of Optelligent, LLC) [25].

and telemetry service module. The SDN controller distributes also configuration commands. Above the SDN controller, the reconfiguration module implements cognitive reconfiguration strategies. In particular, the reconfiguration module uses the SL module and the traditional analytical tools to facilitate the decision making process. The reconfiguration manager estimates the traffic demands between the ToRs by constantly accessing the traffic engineering database. Each time the new estimated traffic matrix arrives, the reconfiguration manager invokes the SL module to determine whether a reconfiguration is required. Let $\mathbb{D}$ denote the set of estimated traffic matrices observed over time and D_t denote an incoming estimated traffic matrix arriving at time t. The SL module firstly leverages an unsupervised learning approach to discover the patterns of the traffic matrices in $\mathbb{D}$ by clustering them into different clusters based on their mutual distance (e.g., Euclidean distance). More specifically, the clusters are formed in a way that the distances between the traffic matrices in the same cluster are less than that in the different clusters [27]. Most of the unsupervised learning algorithms exploit the distance as a similarity measure to compute the similarity between all pairs of data points [28]. Therefore, each traffic matrix in the same cluster has greater similarity compared to those in other clusters. In this work, the SL module adopts the density-based clustering algorithm [29] to cluster the data based on its ability to discover clusters of arbitrary shapes. Because the traffic matrices that belong to the same cluster share a lot of similarities, we make use of the connectivity graph & routing optimization module to determine a common configuration scheme for each cluster. In other words, the reconfiguration manager will only trigger reconfiguration when the currently estimated traffic matrix (D_t) is clustered into a cluster different from the one the previous matrix (D_{t-1}) belongs to. Thus, by exploiting the inherent structure of traffic data, we can largely avoid unnecessary reconfiguration operations while sustaining the desired performance gains.

To determine which cluster a new estimated traffic matrix belongs to, the unsupervised learning module needs to traverse the whole data set, which can be computationally costly when the size of $\mathbb{D}$ is large. To this end, we further introduce a SL mechanism by transferring the pattern learned by unsupervised learning to a supervised data classifier. Specifically, we design a support vector machine (SVM) classifier which takes D_t as input and outputs the predicted cluster ID. Once trained, the prediction complexity of the SVM classifier only depends on the choices of the kernel and is typically proportional to the number of support vectors. Thus, a scalable traffic matrix classification can be achieved. For traffic matrices that can not be classified into any identified clusters, the reconfiguration module instructs the SDN controller to maintain all-to-all interconnection for the related ToRs.

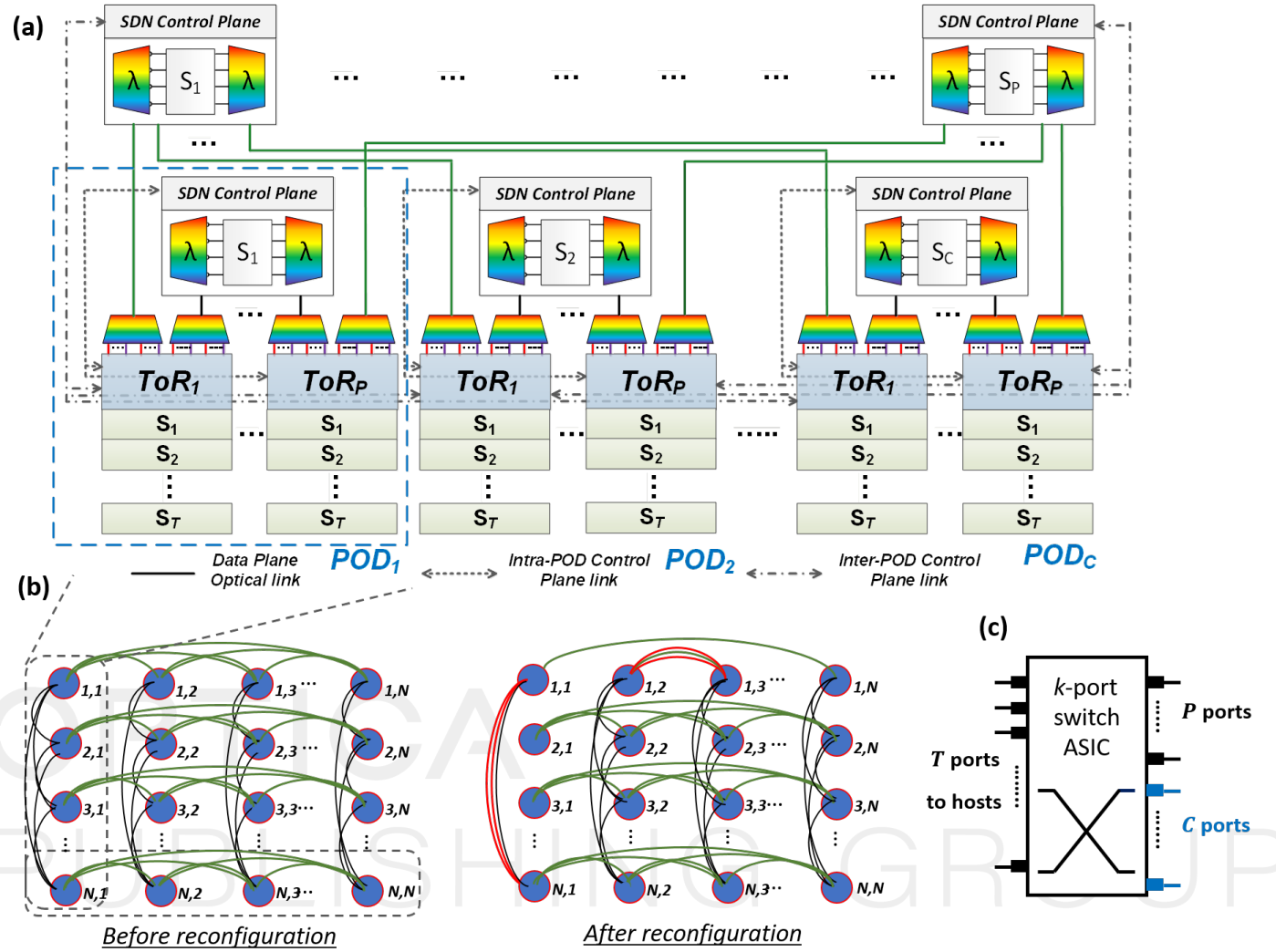

Fig. 2. (a) SL-Hyper-FleX interconnect architecture. (b) PODs are organized into rows and columns. Each column POD has P ToRs. Each row POD has C *ToRs*. (c) k port ToR switch ASIC with T ports for connections to servers, P ports for intra-column POD communication, and C ports for intra-row POD communication. [5]

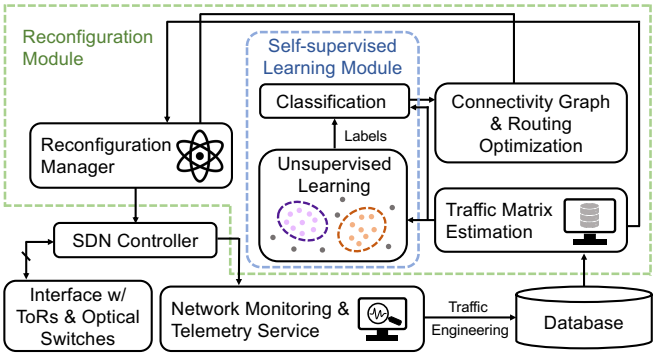

Fig. 3. SDN control plane with reconfiguration functionality

B. Connectivity Graph & Routing Optimization

Next, we present the design of the connectivity graph & routing optimization module used to calculate the graph for a specific communication pattern running in a POD. A more comprehensive design targeting reconfiguration across multiple PODs will be left as future work.

Algorithm 1 summarizes the procedures of calculating a connectivity graph G for a set of N ToRs (T) and a cluster of traffic denoted by $\mathbb{C}_i$. In *Lines* 3-4, for each ToR, we initialize the numbers of available input (x_s) and output ports (y_s) to be $N-1$ and create an empty graph. The basic idea is to iteratively interconnect ToR ports, encouraging steering bandwidth to ToR pairs with larger amounts of traffic pending to be provisioned and larger numbers of ports not assigned yet. We calculate the mean of $\mathbb{C}_i$ and use it as the reference traffic matrix for the cluster (*Line* 5). In *Lines* 6-19, we traverse each ToR by $N-1$ times and attempt to add links to ToR pairs with larger amounts of traffic to be served. Within the loop, we first set r_s as zero for all ToRs to indicate that the corresponding ports of these ToRs have not yet been used (*Lines* 7). Then, with the loop from *Line* 8 to 18, we determine for each ToR s a target ToR (μ^*) to whom the current port should be connected. Specifically, for each other ToR μ, we calculate the products of traffic demand and number of available ports in both directions (i.e., for s to μ and from μ to s) and set the larger value of them as the weight of μ (*Lines* 9). Here, we exclude ToRs whose ports have all been configured by assigning weights of negative infinite. Afterward, in *Lines* 11-12, the algorithm picks ToR u^* with the largest weight as the target ToR and adds a link in both directions to assure bidirectional communications. Finally, *Lines* 13-16 update the information

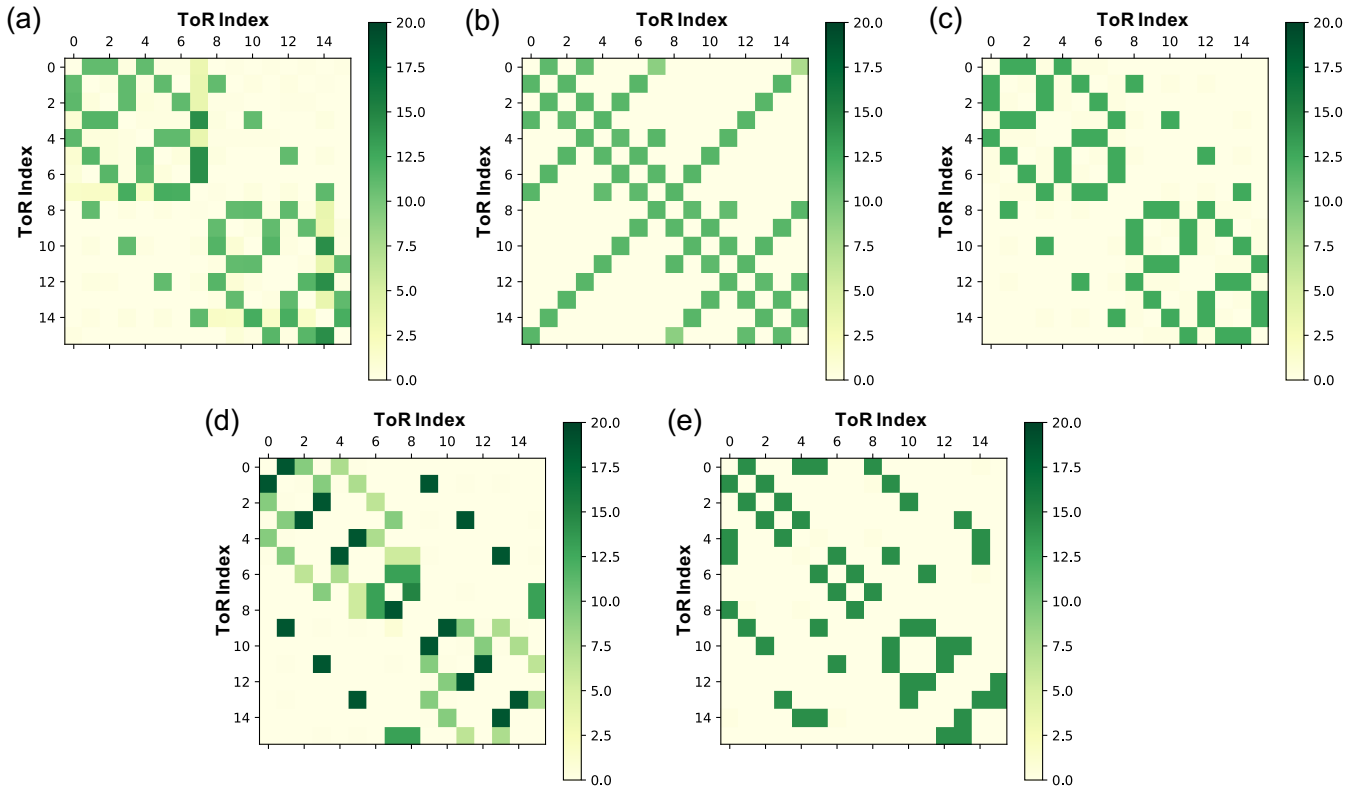

Fig. 4. Traffic matrices used for performance evaluation: (a) Mini AMR, (b) Crystal Router, (c) Fill Boundary, (d) MultiFE, and (e) MultiGrid.

Algorithm 1. Procedures of calculating connectivity graphs.

Input: $T, N, \mathbf{C}_i$
Output: G
$x_s \leftarrow N-1, y_s \leftarrow N-1, \forall s \in T$
$G_{\mu,\nu} \leftarrow 0, \forall \mu, \nu \in T$
$D \leftarrow \frac{1}{|\mathbf{C}_i|} \sum_{D_t \in \mathbf{C}_i} D_t, W \leftarrow D$
for $i \in [1, N-1]$ **do**
  $r_s \leftarrow 0, \forall s \in T$
  for each ToR $s \in T$ **do**

$$w_\mu \leftarrow \begin{cases} \max \begin{Bmatrix} W_{s,\mu} \cdot x_s \\ W_{\mu,s} \cdot y_\mu \end{Bmatrix}, x_s \cdot y_\mu > 0 \\ -Inf, else \end{cases}, \forall \mu \in T \setminus s$$

    if $r_s == 0$ AND $\max_\mu w_\mu > 0$ **then**
      $\mu^* \leftarrow \arg\max_\mu(w_\mu)$
      $G_{s,\mu^*} \leftarrow G_{s,\mu^*} + 1, G_\mu \leftarrow G_{\mu^*,s} + 1$
      $x_{\mu^*} \leftarrow x_{\mu^*} - 1, y_{\mu^*} \leftarrow y_{\mu^*} - 1$
      $x_s \leftarrow x_s - 1, y_s \leftarrow y_s - 1$
      $W_{s,\mu^*} \leftarrow W_{s,\mu^*} - C, W_{\mu^*,s} \leftarrow W_{\mu^*,s} - C$
      $r_{\mu^*} \leftarrow 1$
    end if
  end for
end for

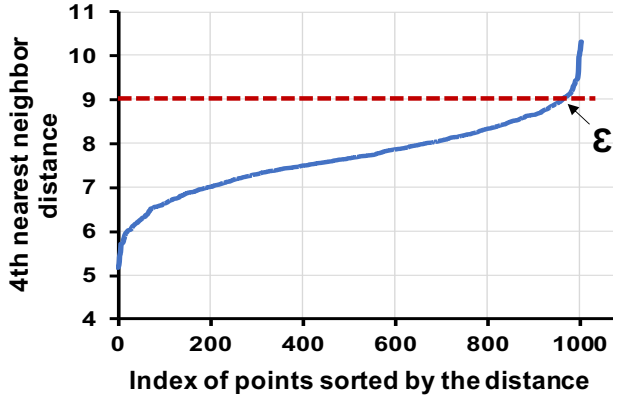

Fig. 5. Distance to the 4^{th} nearest neighbor for each point with respect to index of points sorted by the distance.

of port utilization and traffic demand remaining to be served. After G has been calculated, we rearrange the link connectivity between ToRs and update the topology in the optical domain. Then, in the electrical domain, we apply the equal-cost multi-path (ECMP) algorithm [30] to determine the routing scheme for each flow for its capability of increasing bandwidth utilization by load-balancing traffic over multiple paths.

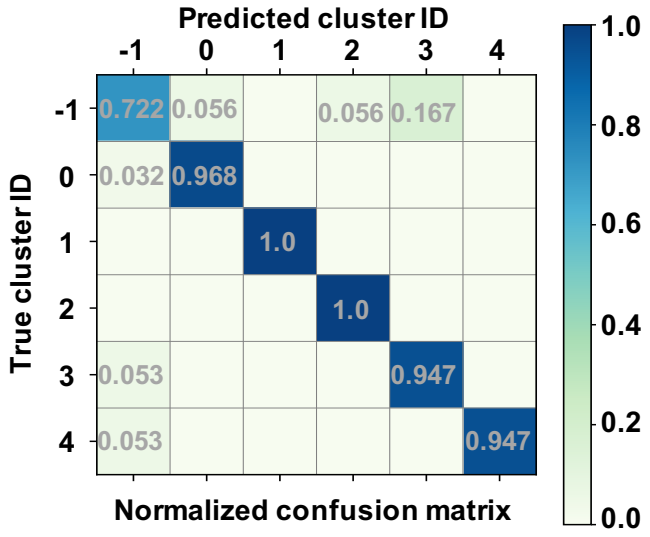

Fig. 6. Classification accuracy of the SVM classifier.

4. PERFORMANCE EVALUATION

A. Prediction Accuracy

We evaluated the proposed cognitive reconfiguration design performance using the Netbench simulator [31] implementing packet-level simulations. A POD of 16 ToRs was considered, where each ToR connects to 16 servers. The servers generate packets following Poisson processes. The upper bound of Alizadeh Web Search distribution [32] was used to emulate the sizes of the flows injected into the network. The source-destination pairs of the packet flows were selected according to the traffic distributions derived from the traffic traces used for the performance evaluation. We assumed an identical wavelength/link capacity of 10 Gbps and a link delay of 20 ns. We adopted data center TCP (DCTCP) [33] as the transport protocol responsible for the communications between two specific network devices. DCTCP is an enhancement of TCP where an explicit congestion notification (ECN) flag is added in the acknowledge packets to provide the record of buffer state along the traversed path. With the assist of ECN, the receivers can reduce their windows before the buffers get full. We set the buffer size of each ToR port as 150,000 Bytes. ECMP routing was used to decide the forwarding of packets. To emulate various traffic patterns, we used a set of traffic matrices from real HPC applications, i.e., Mini Adaptive Mesh Refinement (Mini AMR), Crystal Router, Fill Boundary, MultiFE, and MultiGrid [13]. Fig. 4 shows the heatmaps of the traffic matrices. We expanded the data set by adding random noise with a uniform distribution over -1 to 1 to each traffic trace and obtained a data set of five clusters (each associates with an application) and 1,005 instances.

We evaluated the accuracy of the proposed SL scheme in detecting traffic patterns. We adopted the density-based clustering algorithm in [29] (i.e., DBSCAN), which employs two key parameters, namely, $MinPts$ and ε, to enforce the number of instances required to form a cluster and the distance threshold in identifying neighboring instances, respectively. According to [34], we set $MinPts = 4$ as twice the dimension of input data (i.e., traffic matrices). Then, we applied the approach in [35] to determine the correct setting of ε. Specifically, we calculated the distance to the 4^{th} nearest neighbor for every instance and plotted the distance values in the ascending order as a function of the data index in Fig. 3. We decided the value with respect to the maximum curvature point (i.e., 9) as the most appropriate setting of ε. This is because a larger value of ε encourages forming clusters and the choice of $\varepsilon = 9$ allows the majority of the data instances to be successfully clustered. With such a setting, the clustering algorithm correctly detects five clusters, which we index by cluster ID of 0 to 4. In particular, 837 data instances are clustered into the five clusters while the remaining instances are marked as outliers and are assigned to a cluster ID of -1. Based on the clustering results, we implemented and trained an SVM classifier with kernel type being rbf. Fig. 4 shows the predictive accuracy of the SVM classifier on the testing data set (15% randomly sampled out of the entire data set) in the form of a normalized confusion matrix. We can see that the SVM classifier can achieve a prediction accuracy of above 94% for the data instances belonging to cluster 0 to cluster 4. For cluster -1, the accuracy drops to 72.2% (5 data instances incorrectly classified out of 18 data instances). This is because the number of outliers for training is relatively small. Potential approaches to improve the prediction accuracy for outliers are estimating model uncertainties during predictions and identifying samples with high uncertainties as also outliers [36] or embedding input data into a latent space and exploring the data distribution in the latent space [37].

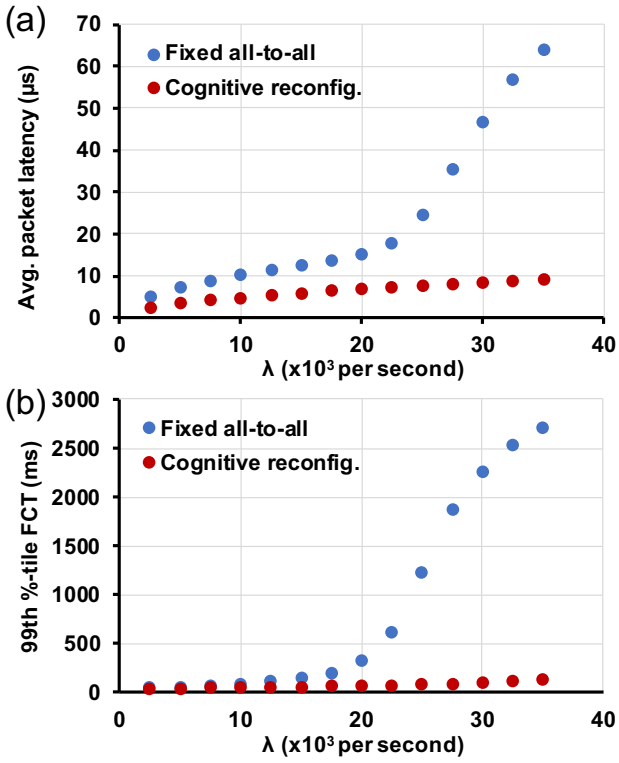

Fig. 7. Comparisons of (a) average packet latency and (b) 99^{th} percentile Flow Completion Time (FCT) between the fixed-all-to-all and the cognitive reconfiguration design.

B. Comparison with Baselines

We first compared the proposed cognitive reconfiguration design with a baseline which always adopting an all-to-all interconnection scheme. Note that the total amount of bandwidth of the

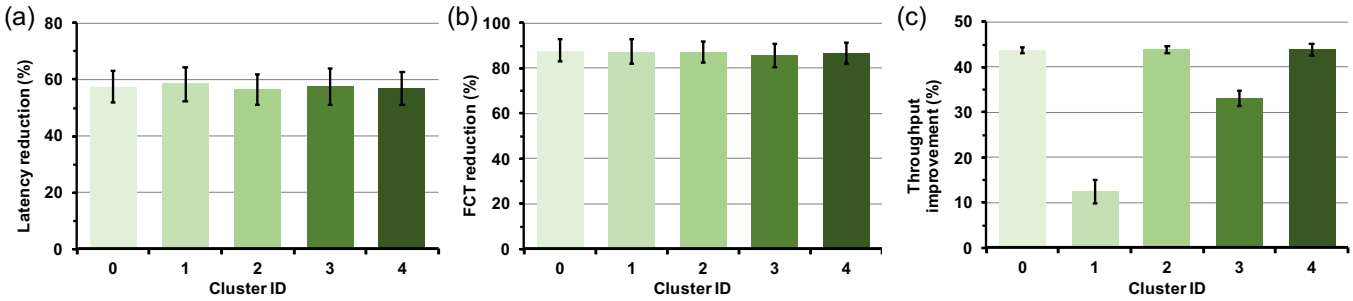

Fig. 8. (a) Latency, (b) FCT, and (c) throughput improvements from the proposed design against the baseline with respect to different clusters.

Table 1. Results of the proposed design against the baseline of reconfiguration using dynamic series with different thresholds

	Reconfiguration w/ different thresholds					Cognitive reconfiguration
	0	15	30	45	60	
Reconfiguration frequency	127	10	10	5	0	12
Latency reduction (%)	56.84 ± 5.89	57.03 ± 5.66	57.03 ± 5.66	42.73 ± 21.80	30.87 ± 23.34	57.06 ± 5.66
FCT reduction (%)	85.79 ± 6.11	86.44 ± 5.59	86.44 ± 5.59	49.76 ± 50.53	-0.73 ± 74.50	85.51 ± 5.55
Throughput improvement (%)	34.69 ± 13.70	36.15 ± 12.03	36.15 ± 12.03	25.11 ± 22.27	6.92 ± 20.94	34.88 ± 13.41

all-to-all interconnection is the same as that of the interconnection after reconfiguration. We perform the comparison in terms of packet latency, flow completion time (FCT), and throughput. Figs. 7(a) and (b) present the results of average packet latency and 99^{th} percentile FCT for the Fill Boundary trace under different network loads λ (average flow arrival rate). The results show that the performance of the two approaches is comparable when the loads are lower than $20,000$ (when the links are largely underutilized). As the load increases, we can see that the performance of the baseline degrades drastically while that of the proposed cognitive reconfiguration design remains stable. This is because the proposed design can effectively steer the bandwidth to where the traffic occurs, reducing the links congestion and increasing the throughput. Such observations also apply to the traffic traces belonging to other clusters. Figs. 8(a)-(c) show the results of improvement from the proposed approach over the baseline in terms of average packet latency, 99^{th} percentile FCT and throughput for each cluster, respectively, when $\lambda = 20,000$. The results show that the proposed design outperforms the baseline for all the clusters.

Next, we assessed the proposed cognitive reconfiguration design under dynamic traffic scenarios. Specifically, the work in [38] suggests that DC traffic patterns are determined by the attributes of jobs executed and typically have the following characteristics: (*i*) the constitution of the jobs is almost invariable and therefore the traffic fluctuates during a period, and (*ii*) the traffic drastically changes at a certain point due to significant differences between job profiles and remains stable afterward. In our simulations, without loss of generality, we considered discrete time intervals and set a fixed period as ten intervals. We specified the traffic pattern for each period by randomly picking a cluster from those recognized and then sampled a traffic matrix from that cluster for each time interval. We generated a sequence of 130 traffic matrices used for evaluation. Note that the proposed approach can dynamically adapt to significant traffic changes between the traffic profiles because the SL module is well-trained with the collected history dataset. When a significant traffic change is detected, the well-trained model can calculate the new configurations corresponding to the new traffic profiles without retraining. The model is periodically retrained when the number of unknown traffic profiles is increasing. We compared our approach with a threshold-based heuristic policy which triggers a reconfiguration operation whenever the difference (i.e., standard deviation) between the current traffic matrix and the one when the reconfiguration was triggered is larger than a predefined threshold. Table. 1 summarizes the results in terms of reconfiguration frequency, latency, FCT, and throughput improvement over the fixed all-to-all approach for the cognitive approach and the threshold-based heuristic with different threshold settings. We can see that the reconfiguration frequency from the heuristic decreases with the increase of threshold, so as the latency, FCT, and throughput performance. The results suggest that it is essential to determine a proper value of threshold (15 in this case) to realize a good trade-off between the number of costly reconfiguration operations and the attainable service performance. Note that the reconfiguration delay is a function of the physical layer switching latency (in the order of microseconds for thermal tuning) and the latency at the control plane layer (to calculate and update the new forwarding tables in the ToR switches). The latter dominates as it is in the order of hundreds of microseconds or even several milliseconds. Given the reconfiguration latency is not negligible and it can cause traffic disruptions, it is important to limit the number of reconfigurations while guaranteeing improvements in terms of

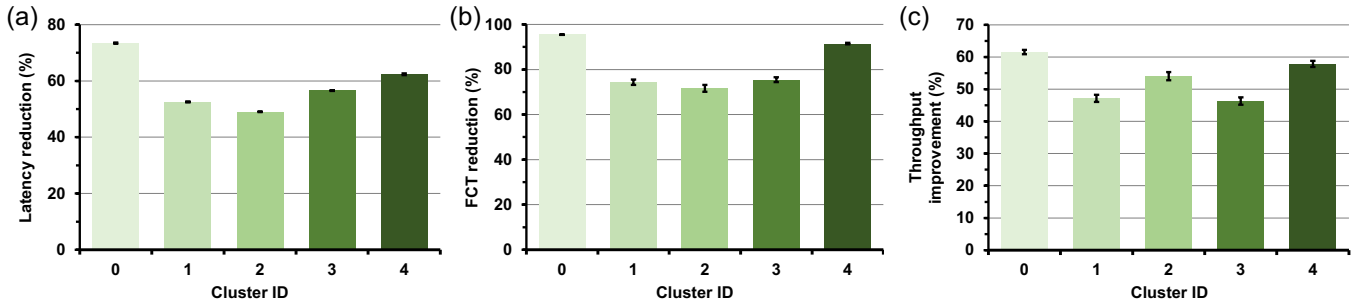

Fig. 9. Results for a POD of 64 ToRs: (a) latency, (b) FCT), and (c) throughput improvements from the proposed design against the baseline with respect to different clusters.

Table 2. Results for a POD of 64 ToRs: the proposed design against the baseline of reconfiguration using dynamic series with different thresholds

	Reconfiguration w/ different thresholds					Cognitive reconfiguration
	0	140	165	195	210	
Reconfiguration frequency	119	11	9	7	0	11
Latency reduction (%)	58.85 ± 7.85	58.87 ± 7.86	54.24 ± 10.19	51.59 ± 15.32	36.89 ± 18.27	58.90 ± 7.91
FCT reduction (%)	81.99 ± 9.51	82.02 ± 9.56	71.13 ± 24.83	69.95 ± 23.86	32.91 ± 43.43	81.98 ± 9.62
Throughput improvement (%)	53.09 ± 5.86	53.16 ± 5.90	44.08 ± 20.77	40.35 ± 21.24	13.76 ± 25.90	53.19 ± 6.03

latency and throughput. However, finding the good trade-off between the frequency of reconfiguration operations and service performance is nontrivial as the threshold-based approach requires many trials and errors to optimize the setting of the threshold in practical network operations. On the other hand, the proposed approach can implement effective reconfiguration strategies by self-learning the characteristics of the traffic data. In Table. 1, it can be seen that the performance of the proposed design is close to that of the heuristic with the best threshold setting.

C. Scalability Analysis

Finally, we investigated the scalability studies of the proposed approach with the simulation based on a POD with 64 ToRs. Based on the parameter setting method discussed in Subsection A, we assigned ε as a value of 56.3. With such a setting, the clustering algorithm successfully detects five clusters and a predictive accuracy of above 94% can be obtained from a well-trained SVM classifier. Figs. 9(a)-(c) present the results of improvement from the proposed approach over the fixed all-to-all approach in terms of average packet latency, 99th percentile FCT and throughput for each cluster, respectively (when $\lambda = 60,000$). For all clusters, we can see that compared with a static all-to-all interconnection, the proposed approach can improve the throughput by up to 1.62× while reducing the latency and FCT by up to 3.84× and 20×, respectively. Table. 2 gives a summary of the results in terms of reconfiguration frequency, latency, FCT, and throughput improvement over the fixed all-to-all approach for the cognitive approach and the threshold-based approach with different threshold settings. It can be seen that the performance of the proposed approach is close to that of the heuristic approach with the best threshold setting (140 in this case), while our approach does not require many trials and errors to achieve such performance. The results confirm the effectiveness of self-supervised learning also under more complex network configurations and can further verify the scalability of self-supervised learning. Meanwhile, we want to emphasize that the complexity of our approach increases linearly with the number of TORs in a POD (i.e., the size of the traffic matrix). Specifically, the time complexity of calculating Euclidean distance in the unsupervised learning module increases linearly with the size of the traffic matrix (i.e., $\mathcal{O}(n)$). The complexity of SVM depends only on support vectors, and recent studies argued that the SVM does not suffer from the high dimensionality of input space [39, 40]. However, recent studies have demonstrated successful model training by unsupervised learning with an input size of more than 200-by-200 [41, 42].

5. CONCLUSION

This paper discusses a flexible-bandwidth photonic interconnect architecture with cognitive reconfiguration to adapt the network to different traffic profiles. We focused particularly on a self-supervised machine learning approach for decision making of bandwidth reconfiguration of the Hyper-FleX links. The simulation results show that our approach achieves significant latency, FCT, and throughput improvements over a regular all-to-all baseline. Note that the co-design of the data plane (hardware) and control plane (software and algorithms) is key for using any optical switching paradigm. As part of our future studies, we will design a routing, bandwidth, and topology

assignment (RBTA) that can leverage the two FSRs discusses above to co-design the optical reconfiguration and packet-level routing in the ToRs (e.g., using weighted-cost multi-path routing [43]) and achieve reconfiguration with minimal or no traffic disruption. Other studies will include testbed demonstrations of co-designed control and data planes leveraging the proposed SL algorithm, extending the reconfiguration scenario also to multiple PODs.

ACKNOWLEDGMENT

This work was supported in part by the NSF ECCS Award #1611560, NSFC Project U2001601, and NSFC Project 62035018.

REFERENCES

1. F. Petrini and M. Vanneschi, "k-ary n-trees: high performance networks for massively parallel architectures," in *Proceedings 11th International Parallel Processing Symposium,* (1997), pp. 87–93.
2. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication,* (Association for Computing Machinery, New York, NY, USA, 2008), SIGCOMM '08, p. 63–74.
3. A. Rahim, T. Spuesens, R. Baets, and W. Bogaerts, "Open-access silicon photonics: Current status and emerging initiatives," Proc. IEEE **106**, 2313–2330 (2018).
4. S. Y. Siew, B. Li, F. Gao, H. Y. Zheng, W. Zhang, P. Guo, S. W. Xie, A. Song, B. Dong, L. W. Luo, C. Li, X. Luo, and G.-Q. Lo, "Review of silicon photonics technology and platform development," J. Light. Technol. **39**, 4374–4389 (2021).
5. G. Liu, R. Proietti, M. Fariborz, P. Fotouhi, X. Xiao, and S. Ben Yoo, "Architecture and performance studies of 3d-hyper-flex-lion for reconfigurable all-to-all hpc networks," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis,* (2020), pp. 1–16.
6. N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," SIGCOMM Comput. Commun. Rev. **40**, 339–350 (2010).
7. H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, and H. Williams, "Sirius: A flat datacenter network with nanosecond optical switching," In Proc. Annu. conference ACM Special Interest Group on Data Commun. on applications, technologies, architectures, protocols for computer communication (2020).
8. X. Guo, F. Yan, J. Wang, G. Exarchakos, Y. Peng, X. Xue, B. Pan, and N. Calabretta, "Rdon: a rack-scale disaggregated data center network based on a distributed fast optical switch," J. Opt. Commun. Netw. **12**, 251–263 (2020).
9. M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," J. Opt. Commun. Netw. **12**, B44–B54 (2020).
10. Z. Cao, R. Proietti, M. Clements, and S. J. B. Yoo, "Experimental demonstration of flexible bandwidth optical data center core network with all-to-all interconnectivity," J. Light. Technol. **33**, 1578–1585 (2015).
11. G. Yuan, R. Proietti, X. Liu, A. Castro, D. Zang, N. Sun, C. Liu, Z. Cao, and S. J. B. Yoo, "Aron: Application-driven reconfigurable optical networking for hpc data centers," in *ECOC 2016; 42nd European Conference on Optical Communication,* (2016), pp. 1–3.
12. X. Chen, R. Proietti, M. Fariborz, C.-Y. Liu, and S. J. B. Yoo, "Machine-learning-aided cognitive reconfiguration for flexible-bandwidth hpc and data center networks [invited]," J. Opt. Commun. Netw. **13**, C10–C20 (2021).
13. "Characterization of DOE Mini-apps," https://portal.nersc.gov/project/CAL/doe-miniapps.htm (2020). Accessed: 11-2020.
14. N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of splash-2 and parsec," in *2009 IEEE International Symposium on Workload Characterization (IISWC),* (2009), pp. 86–97.
15. M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, and J. Zhu, "Neural network meets dcn: Traffic-driven topology adaptation with deep learning," Proc. ACM Meas. Anal. Comput. Syst. **2** (2018).
16. J. Guo and Z. Zhu, "When deep learning meets inter-datacenter optical network management: Advantages and vulnerabilities," J. Light. Technol. **36**, 4761–4773 (2018).
17. S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "Deepconf: Automating data center network topologies management with machine learning," in *Proceedings of the 2018 Workshop on Network Meets AI & ML,* (Association for Computing Machinery, New York, NY, USA, 2018), NetAI'18, p. 8–14.
18. H. Fang, W. Lu, Q. Li, J. Kong, L. Liang, B. Kong, and Z. Zhu, "Predictive analytics based knowledge-defined orchestration in a hybrid optical/electrical datacenter network testbed," J. Light. Technol. **37**, 4921–4934 (2019).
19. X. Guo, F. Yan, X. Xue, B. Pan, G. Exarchakos, and N. Calabretta, "Qos-aware data center network reconfiguration method based on deep reinforcement learning," J. Opt. Commun. Netw. **13**, 94–107 (2021).
20. Y. Shang, X. Chen, R. Proietti, B. Guo, S. Huang, and S. J. B. Yoo, "Deepautonet: Self-driving reconfigurable hpc system with deep reinforcement learning," in *Asia Communications and Photonics Conference (ACPC) 2019,* (2019), p. S3C.4.
21. X. Xue, F. Wang, F. Agraz, A. Pages, B. Pan, F. Yan, S. Spadaro, and N. Calabretta, "Experimental assessment of sdn-enabled reconfigurable opsquare data center networks with qos guarantees," in *2019 Optical Fiber Communications Conference and Exhibition (OFC),* (2019), pp. 1–3.
22. C. Liu, M. Xu, and S. Subramaniam, "A reconfigurable high-performance optical data center architecture," in *2016 IEEE Global Communications Conference (GLOBECOM),* (2016), pp. 1–6.
23. R. Proietti, Z. Liu, X. Xiao, X. Chen, and S. J. B. Yoo, "Energy-efficient and scalable data centers with flexible bandwidth siph all-to-all fabrics," in *2021 Optical Fiber Communications Conference and Exposition (OFC),* (2021), pp. 1–3.
24. J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis,* (2009), pp. 1–11.
25. X. Xiao, R. Proietti, G. Liu, H. Lu, Y. Zhang, and S. J. B. Yoo, "Multi-fsr silicon photonic flex-lions module for bandwidth-reconfigurable all-to-all optical interconnects," J. Light. Technol. **38**, 3200–3208 (2020).
26. Y. Shen, M. H. N. Hattink, P. Samadi, Q. Cheng, Z. Hu, A. Gazman, and K. Bergman, "Software-defined networking control plane for seamless integration of multiple silicon photonic switches in datacom networks," Opt. Express **26**, 10914–10929 (2018).
27. O. Aouedi, S. Hamma, J. K. M. Perera, and K. Piamrat, "Network Traffic Analysis using Machine Learning: an unsupervised approach to understand and slice your network," Annals of Telecommunications - annales des t@ACUTEACCENT@el@ACUTEACCENT@ecommunications (2021).
28. J. Irani, N. Pise, and M. Phatak, "Clustering techniques and the similarity measures used in clustering: A survey," Int. J. Comput. Appl. **134**, 9–14 (2016).
29. M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of KDD,* (1996), pp. 226–231.
30. C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992 (2000).
31. S. Kassing, A. Valadarsky, and A. Singla, "Netbench," https://github.com/ndal-eth/netbench (2016).
32. M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Pfabric: Minimal near-optimal datacenter transport," SIGCOMM Comput. Commun. Rev. **43**, 435–446 (2013).
33. M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," SIGCOMM Comput. Commun. Rev. **40**, 63–74 (2010).
34. J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," Data

Min. Knowl. Discov. **2**, 169–194 (1998).
35. X. Chen, B. Li, R. Proietti, Z. Zhu, and S. J. B. Yoo, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," J. Light. Technol. **37**, 1742–1749 (2019).
36. Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning,* vol. 48 of *Proceedings of Machine Learning Research* M. F. Balcan and K. Q. Weinberger, eds. (PMLR, New York, New York, USA, 2016), pp. 1050–1059.
37. H. Lun, X. Liu, M. Cai, Y. Wu, M. Fu, L. Yi, W. Hu, and Q. Zhuge, "Gan based soft failure detection and identification for long-haul coherent transmission systems," in *2021 Optical Fiber Communications Conference and Exhibition (OFC),* (2021), pp. 1–3.
38. D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication,* (Association for Computing Machinery, New York, NY, USA, 2012), SIGCOMM '12, p. 199–210.
39. C. Cortes and V. N. Vapnik, "Support-vector networks," Mach. Learn. **20**, 273–297 (2004).
40. J. A. Gualtieri, "The support vector machine (svm) algorithm for supervised classification of hyperspectral remote sensing data," Kernel methods for remote sensing data analysis **3**, 51–83 (2009).
41. Q. V. Le, "Building high-level features using large scale unsupervised learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing,* (2013), pp. 8595–8598.
42. M. Leordeanu, R. Collins, and M. Hebert, "Unsupervised learning of object features from video sequences," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1 (IEEE Computer Society; 1999, 2005), p. 1142.
43. J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "Wcmp: Weighted cost multipathing for improved fairness in data centers," in *Proceedings of the Ninth European Conference on Computer Systems,* (Association for Computing Machinery, New York, NY, USA, 2014), EuroSys '14.