Machine Learning for Computer Systems and Networking: A Survey

MARIOS EVANGELOS KANAKIS, Vrije Universiteit Amsterdam
RAMIN KHALILI, Huawei Munich Research Center
LIN WANG, Vrije Universiteit Amsterdam and TU Darmstadt

Machine learning (ML) has become the de-facto approach for various scientific domains such as computer vision and natural language processing. Despite recent breakthroughs, machine learning has only made its way into the fundamental challenges in computer systems and networking recently. This article attempts to shed light on recent literature that appeals for machine learning-based solutions to traditional problems in computer systems and networking. To this end, we first introduce a taxonomy based on a set of major research problem domains. Then, we present a comprehensive review per domain, where we compare the traditional approaches against the machine learning-based ones. Finally, we discuss the general limitations of machine learning for computer systems and networking, including lack of training data, training overhead, real-time performance, and explainability, and reveal future research directions targeting these limitations.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computer systems organization**; • **Networks**;

Additional Key Words and Phrases: Machine learning, computer systems, computer networking

ACM Reference format:
Marios Evangelos Kanakis, Ramin Khalili, and Lin Wang. 2022. Machine Learning for Computer Systems and Networking: A Survey. *ACM Comput. Surv.* 55, 4, Article 71 (November 2022), 36 pages.
https://doi.org/10.1145/3523057

1 INTRODUCTION

Revolutionary research in **machine learning** (**ML**) has significantly disrupted the scientific community by contributing solutions to long-lived challenges. Thanks to the continuous advancements in computing resources (e.g., cloud data centers) and performance capabilities of processing units (e.g., accelerators like GPUs and TPUs), ML, particularly its rather computation-expensive subset namely **deep learning** (**DL**), has gained its traction [120, 131]. In general, ML has established dominance in vision tasks such as image classification, object recognition [86], and more to follow [58, 156]. Other remarkable examples where ML is thriving include speech recognition [52]

This work has been partially funded by the Dutch Research Council (NWO) Open Competition Domain Science XS Grant 12611 and by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1053 MAKI.
Authors' addresses: M. E. Kanakis, Vrije Universiteit Amsterdam, De Boelelaan 1111, Amsterdam, The Netherlands; email: marioskanakis@gmail.com; R. Khalili, Huawei Munich Research Center, Riesstraße 12, Munich, Germany; email: ramin.khalili@huawei.com; L. Wang (corresponding author), Vrije Universiteit Amsterdam, De Boelelaan 1111, Amsterdam, The Netherlands and TU Darmstadt, Hochschulstraße 10, Darmstadt, Germany; email: lin.wang@vu.nl.

0360-0300/2022/11-ART71 $15.00
https://doi.org/10.1145/3523057

and machine translation [155]. ML is also prevailing to a plethora of specialized tasks that prior work has been far out of reach to yield notable outcomes [2, 141]. For instance, it was not until recently that top professional Go players were beaten by a **deep reinforcement learning** (**DRL**) agent [141].

Considering this unprecedented growth of ML in various classification/control tasks, one begs the question, how can we apply ML to other domains that have long suffered from a sub-optimal performance that traditional solutions can offer at their best? One prominent example is the domain of computer systems and networking, where parameter tuning and performance optimization largely rely on domain expertise and highly-engineered heuristics. Essentially, it is of great interest to answer whether it is time to make machines learn to optimize their performance by themselves automatically. Putting it into perspective, there is a multitude of challenges for which ML can prove beneficial due to its innate ability to capture complex properties and extract valuable information that no human, even the domain expert, can master.

We observe two general challenges in the current research practice of applying ML in computer systems and networking. First, there is no consensus on a common guideline for using ML in computer systems and networking (e.g., when ML would be preferable over traditional approaches), with research efforts, so far, scattered in different research areas. The lack of a holistic view makes researchers difficult to gain insights or borrow ideas from related areas. Second, there have not been any recent efforts that showcase how to select an appropriate ML technique for each distinct problem in computer systems and networking. In particular, we observe that in some cases, only a certain ML algorithm is suitable for a given problem, while there exist also problems that can be tackled through a variety of ML techniques and it is nontrivial to choose the best one. The above challenges constitute major obstacles for researchers to capture and evaluate recent work sufficiently, when probing for a new research direction or optimizing an existing approach.

In this article, we tackle these challenges by providing a comprehensive *horizontal* overview to the community. We focus on the research areas of computer systems and networking, which share similar flavor and have seen promising results through using ML recently. Instead of diving into one specific, vertical domain, we seek to provide a cross-cutting view for the broad landscape of the computer systems and networking field. Specifically, we make the following contributions:

— We present a taxonomy for ML for computer systems and networking, where representative works are classified according to the taxonomy.
— For each research domain covered in the taxonomy, we summarize traditional approaches and their limitations, and discuss ML-based approaches together with their pros and cons.
— We discuss the common limitations of ML when applied in computer systems and networking in general and reveal future directions to explore.

With these contributions, we expect to make the following impact: (1) introducing new researchers having no domain expertise to the broad field of ML for systems and networking, (2) bringing awareness to researchers in certain domains about the developments of applying ML on problems in neighboring domains and enabling to share and borrow ideas from each other.

Related surveys. There has not been a survey that satisfies the objectives we aim at achieving in this article. Most of the related surveys are domain-specific, focusing on a narrow vertical overview. For example, Zhang et al. present a survey on leveraging DL in mobile and wireless networking [195]; hence, we will skip these areas in this survey. There are also surveys focusing on areas like compiler autotuning [6], edge computing [37], and Internet-of-Things [63]. While targeting different levels of concerns, these surveys can facilitate domain experts to gain a deep understanding of all the technical details when applying ML on problems from the specific domain. However, they miss the opportunity to show the broad research landscape of using ML in

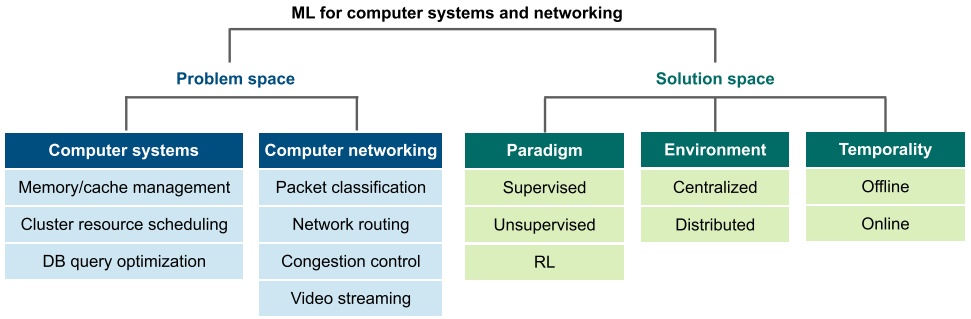

Fig. 1. A taxonomy of ML for computer systems and networking.

the general computer systems and networking field. We aim at bridging such a gap in this work. The closest work to ours is [174]. Focusing on networking, this survey provides an overview of ML applied in networking problems but ignores the computer systems part. Besides, the article was published almost four years ago. Considering that significant progress has been made in recent years, we believe it is time to revisit this topic.

2 TAXONOMY

We first categorize the existing work on ML for computer systems and networking. Figure 1 presents a taxonomy from two angles: the *problem* space and the *solution* space. The problem space covers fundamental problems that have been extensively studied in the traditional computer systems and networking research area. The solution space is constructed based on our experience, where the most important feature dimensions of ML-based solutions are included. We highlight over 150 articles proposed for the problems falling in the taxonomy where ML is mentioned. The article selection is based on a comprehensive approach to cover as broadly as possible the articles in each of the selected research domains. For each of the covered domains, we selectively pick the more notable works and we provide more elaboration on their contributions.

2.1 Problem Space

In the problem space, we focus our attention on representative research problems from both the computer systems and networking communities. These problems are selected based on the following principles: (1) The problem should be fundamental in the considered domain, not a niche area that requires heavy background knowledge to understand. (2) There should still be active research efforts made on addressing the problem. (3) There should be considerable research on applying ML to tackle the problem. For computer systems, we will look at three problems: memory/cache management, cluster resource scheduling, and query optimization in databases. For computer networking, we will focus on four problems: packet classification, network routing, congestion control, and video streaming. This categorization helps the readers (1) dive into the specific topics of interest directly, and (2) obtain an overview of the other problems in the neighboring fields that have benefited from ML as well. Here, we provide a brief introduction to these problems:

Computer systems. A computer system is broadly defined as a set of integrated computing devices that take, generate, process, and store data and information. Generally speaking, a computer system is built with at least one computing device. In the literature, both single-device computer systems as well as distributed systems consisting of a set of computing devices have been extensively explored. The research goals in computer systems include performance, energy efficiency,

reliability, and security. In this survey, we focus on three fundamental problems in computer systems, each representing one level of the system abstractions:

— Memory/cache management is a representative decision-making problem domain at the level of single-device operating systems. The main problems include memory prefetching from DRAM to CPU cache and page scheduling from disk to DRAM.
— Cluster resource scheduling is a core task at the level of distributed computing infrastructure, which concerns the allocation of cluster resources to computing jobs in a distributed setting, meeting set goals including resource efficiency, job completion time, among others.
— Query optimization is a central problem in databases—a representative application in systems. Given a query, the problem is to find the most efficient query execution plan.

Computer networking. A computer network is an interconnection of multiple computing devices (a.k.a. hosts) where data can be sent/received among these connected devices. Apart from the hosts, computer networks involve devices that are responsible for forwarding data between hosts, which are called network devices including routers and switches. Computer networking is a long-lasting research domain where we have seen a significant number of artifacts and control mechanisms. In particular, we will look at the following four fundamental problems in networking ranging from packet-level to connection-level, and to application-level:

— Packet classification is a basic networking functionality in almost all network devices. The problem of network packet classification is to decide the category of packets according to some predefined criteria with high efficiency (high speed, low resource footprint).
— Network routing concerns finding the best path for delivering packets on a network, given some performance metrics such as latency.
— Congestion control is a network mechanism in the transport layer to provide connection-oriented services based on best-effort network delivery.
— Video streaming is one of the most popular network applications, which is mostly based on the concept of **adaptive bitrate** (**ABR**) nowadays. ABR aims at choosing the most suitable bitrate for delivering video segments under dynamic network conditions.

2.2 Solution Space

Existing work of applying ML for computer systems and networking problems can also be viewed from the angle of the solution space, namely which learning paradigm/algorithm is applied.

Learning paradigm. There are generally three types of learning paradigms, namely supervised learning, unsupervised learning, and **reinforcement learning** (**RL**), and all of them have been applied to some of the problems we cover in this survey. We refer readers not familiar with these paradigms to a general introduction in [87].

Environment. There are generally two types of environments our considered problems can be in: centralized and distributed. A centralized environment involves a single entity where decision-making is based on global information, while distributed environments involve multiple possibly coordinated autonomous entities. While it is natural to have a distributed solution in a distributed environment, distributed learning is generally more difficult than centralized ones, mainly due to the limitations in coordination. **Federated learning** (**FL**) is a distributed learning technique, where the client workers perform the training and communicate with a central server to share the trained model instead of the raw data [7]. Multi-agent techniques [121] are other examples of distributed learning. In this survey, We categorize the solutions based on whether they are centralized or distributed, but we do not go further into details of the learning technique (e.g., FL). For more information about FL systems, we refer the reader to recently published surveys such as [190].

Table 1. Summary of Selected ML Solutions for Computer Systems and Networking Problems

Solution	Paradigms			Environment		Temporality	
	SL	USL	RL	Centralized	Distributed	Offline	Online
Memory/cache management							
LSTM Hardware Prefetcher [194]	✓			✓	✓	✓	
Learning Access Patterns [57]	✓	✓		✓	✓	✓	
Compact Prefetcher [149]	✓			✓			✓
Kleio [36]	✓			✓	✓	✓	
Lightweight Caching [12]	✓			✓		✓	
RL-Cache [80]			✓	✓		✓	
Cluster resource scheduling							
DeepRM [106]			✓	✓		✓	
Device Placement [118]			✓		✓		✓
Decima [108]			✓	✓		✓	
Query optimization in databases							
Learned Index Structures [84]	✓			✓		✓	
SkinnerDb [162]			✓	✓			✓
DQ [85]			✓	✓			✓
State Representations [125]			✓	✓		✓	
MSCN [79]	✓			✓		✓	
Neo [110]			✓	✓			✓
Packet classification							
Deep Packet [100]	✓			✓		✓	
NeuroCuts [97]			✓	✓		✓	
Network routing							
Learning to Route [165]			✓	✓		✓	
DQRC [191]			✓		✓		✓
Congestion control							
Remy [179]			✓	✓		✓	
Vivace [35]	✓			✓		✓	
Aurora [71]			✓	✓		✓	
Orca [1]			✓		✓		✓
DRL-CC [184]			✓	✓		✓	
Video streaming							
CS2P [154]		✓		✓		✓	
Pensieve [107]			✓	✓		✓	

SL: supervised learning, USL: unsupervised learning, RL: reinforcement learning.

Temporality. Learning can also be divided into two fashions with respect to their temporality: offline and online. Offline learning requires to pre-train an ML model with existing data in advance and the trained model is applied in decision making without being trained on further input experience. Online learning involves the continuous learning of ML models, where at inference time, the model is also updated after experiencing the given input. Depending on the scenario, a model may first be trained offline and then re-trained online.

2.3 Classification of Selected Works

Before we dive into each of the problem domains in the following sections, we provide a cross-cutting view for all the fields, covering the major works and showing how they can be classified with respect to the solution space described above. Such a view is provided in Table 1.

3 MEMORY/CACHE MANAGEMENT

Typical state-of-the-art computer systems utilize multi-layered memory devices and involve several complex memory management operations. Despite the technological advancements, i.e., the

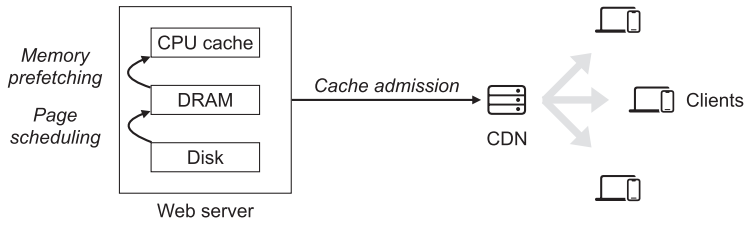

Fig. 2. Representative memory/cache management problems: memory prefetching and page scheduling in operating systems, and cache admission in content delivery networks (CDNs).

exponential reduction of storage cost over the decades and the inverse expansion of the size, storage systems remain the untamed stallion of performance bottlenecks in every computer system.

Figure 2 depicts the major storage-related problems in existing computer systems. In the grand scheme of memory operations, retrieving an entry from the CPU cache is a matter of nanoseconds, but conditionally advances by several orders of magnitude when a cache miss occurs and the entry must be fetched from DRAM or a disk. Over the years, a significant amount of work has been dedicated to tackling the inefficiencies and induced latency of traversing the various memory hierarchy. In several cases, sophisticated mechanisms have been proposed that concern preemptive actions. In other words, there is ongoing research on how to prefetch data or instructions from DRAM to the CPU cache and schedule hot pages from the disk to DRAM. At the networked system level, complex cache admission and invalidation policies for **content delivery networks (CDNs)** have also been explored with the aim of performance optimization in delivering large content such as video data. The common challenge in all these storage-related problems consists in the prediction of the data pattern in these systems, which has become increasingly challenging due to the growing complexity in how applications access their data.

3.1 Traditional Approaches and Limitations

A significant amount of research has been focused on mitigating memory-induced bottlenecks. One example is *memory prefetching*, which preloads memory content into the register or cache to avoid slow memory access. There are generally two ways to implement a memory prefetcher: software-based and hardware-based. Software-based prefetchers use explicit instructions for prefetching. While offering flexibility, software-based prefetchers suffer from increased code footprint and latency, and low accuracy. Therefore, mainstream memory prefetchers are implemented in hardware integrated in the CPU. State-of-the-art hardware prefetchers typically rely on CPU's memory access pattern and compute a corresponding delta based on the access pattern for prefetching [67, 116, 123, 140, 144, 192]. However, prefetchers of this type become sub-optimal when memory accesses are highly irregular. On the other hand, prefetchers based on pattern history perform much better at capturing irregularities but are more expensive to integrate [194].

Page scheduling aims at improving performance by providing pages that are frequently accessed close to the computing units such as DRAM. Page scheduling exhibits high complexity and vast research efforts have attempted to address it thoroughly [22, 38, 73, 114, 139, 181, 182]. Common approaches on page scheduling usually involve system-level integration, for example, in the operating system or during compilation. Current state-of-the-art leverages history information to predict future memory accesses. Yet, the performance bottleneck still exists [36].

Towards caching on a larger scale, CDNs focus on optimizing the latency for content requested from users [80]. To do so, CDNs utilize *cache admission and eviction* policies that fetch and remove

objects from the cache, respectively. Whenever a requested object is not in the cache and has to be fetched, the user suffers from degraded performance [80]. Extended research has been conducted on admission/eviction schemes for cache-miss optimization [23, 39, 80, 102].

Limitations of traditional approaches: Rule-based solutions are often sub-optimal since the data pattern is too complex to specify with simple rules. On the other hand, sophisticated solutions that explore the deep spatial and temporal correlations in the data are hard to make their way into real-world systems since they are expensive to implement and have poor generality when facing different applications. Such contradiction has led to almost stagnant developments in memory prefetching and caching solutions.

3.2 ML-based Approaches

As mentioned above, data pattern prediction is of paramount importance for prefetching and caching algorithms. ML, by its nature, is a powerful tool to explore hidden patterns in irregular data. Thus, the use of ML to these storage-related problems is well justified.

3.2.1 Memory Prefetchers. A notable work in memory prefetchers with neural networks is introduced in [194]. The authors target pattern history-based prefetchers, and more specifically **variable length delta prefetcher** (**VLDP**) [140]. The authors leverage **long short-term memory** (**LSTM**), an **recurrent neural networks** (**RNNs**) based learning algorithm, to predict the memory addresses of upcoming accesses to memory. The authors integrate the LSTM neural network in the last cache level to make predictions in a bounded environment—the OS page [194]. Implementation and evaluation are conducted in an offline manner, where the prefetcher is tested on accuracy and coverage. Another work formulates prefetching as a classification problem and proposes two variants based on embedding LSTM and a hybrid approach respectively [57]. With embedding LSTM a small constant number of predictions per time step are performed in both a local and global context. In the hybrid approach, k-means clustering is used to partition the memory space into regions, and then a neural network is used for inference in each region. Srivastava et al. target the limitations of prior ML-based approaches and present a more robust solution for integrating a learning-based predictor to current system architectures. Similar to previous works, the authors use the LSTM to predict future memory accesses. Besides building a model with high accuracy, the authors employ model compression to increase the inference speed and achieve substantial performance gains in execution. Further, they propose to learn a policy online to retrain the model when the accuracy drops below a predefined threshold [149]. This enables the proposed solution to adjust to real-world environments where a specialized approach might not fit the mould. Overall, results are promising against traditional prefetchers and unfold a new step towards learned memory management systems.

3.2.2 Page Scheduling. Kleio leverages LSTM neural networks to predict page access counts in applications that heavily impact the system's performance [36]. Kleio identifies crucial pages that will increase the application's performance and trains an LSTM network for each. This approach significantly boosts accuracy as each network is able to capture naturally the problem space of page scheduling, and the neural networks come greatly reduced in terms of output range values, which contribute to overall better predictions. Additionally, pages that are not crucial for the system's performance fall back to the existing history-based page scheduler [36]. Evaluation indicates that

Kleio's neural network predictions sharply enhance the application's performance, while accuracy indicators expose severe limitations of history-based schedulers.

3.2.3 Cache Admission and Eviction in CDNs. Recently, two remarkable contributions [12, 80] leverage ML techniques for cache admission policies in CDNs. Berger proposes a supervised learning scheme based on optimal caching decisions (OPT) [12]. The proposed scheme, LFO, learns a caching policy that maps features to those of OPT, essentially predicting whether an object should be admitted to the cache [12]. LFO achieves high accuracy with negligible delay, constituting a feasible alternative for production. While Berger advocates against RL for cache admission due to increased complexity and slow reward convergence [12], RL-Cache leverages RL to optimize directly for hit rate [80]. Based on a complete set of features, RL-Cache trains a neural network that decides upon an object request, whether it is to be admitted to the cache or not [80]. RL-Cache trains on trace requests from production CDNs and employs LRU as its eviction policy. Additionally, RL-Cache is optimized for deployment, considering that periodic training can be incurred in a different location relieving the content server. RL-Cache competes seriously with state-of-the-art schemes and composes an interesting direction for further research. Fedchenko et al. take a different approach on content caching with ML in [45]. Instead of utilizing LSTM neural networks for the prediction of sequences or time series as explained above, a simple feed-forward neural network is used to predict the most popular entries. However, performance advancements are insignificant when compared with existing policies [149], while the authors report that treating the problem as classification, similarly to [57], is a direction worth exploring [45].

3.3 Discussion on ML-based Approaches

A common theme among most existing work is the use of RNNs and in particular LSTM neural networks. This strikes as the de-facto consideration when it comes to memory-related challenges. The ability of RNNs to preserve state is what makes them powerful in problems involving predictions on sequences of data or data in a time series. This is in clear contrast to traditional approaches, which suffer poor predictions on the complex data pattern. Meanwhile, ML-based approaches have become more accessible due to various AutoML solutions and tools. Another similar trait lies in the selection of learning algorithms the authors have to make. We observe that most approaches rely on supervised learning and making predictions. Considering the nature of the problems they are targeting, this also comes naturally. Yet, we observe that despite the fitting-the-mould type of approach that researchers follow at the infant stage when results are not significant, many authors attempt to solve the problem with an unorthodox methodology. For instance, RL-Cache leverages RL to construct a cache admission policy rather than following a statistical estimation approach [80]. However, this does not always translate to successful solutions, albeit it certainly denotes a pattern on how researchers apply learned solutions to traditional and emerging challenges. Overall, ML-based approaches have demonstrated their clear benefits for problems in memory systems when facing complex data patterns and provide multiple easy-to-generalize techniques to tackle these problems from different angles. However, if the underlying data pattern is simple and easy to obtain, using ML-based approaches would become an overkill.

4 CLUSTER RESOURCE SCHEDULING

Resource scheduling concerns the problem of mapping resource demands to computing resources meeting set goals on resource utilization, response time, fairness, and affinity/anti-affinity constraints. Cloud-based solutions nowadays dominate the computing landscape, providing high

scalability, availability, and cost efficiency. Scheduling in the cloud environment goes beyond a single or multi-core computing node and needs to deal with a multitude of physical nodes, sometimes also equipped with heterogeneous domain-specific accelerators. The scope of cloud resource scheduling can be within a single cloud data center or across geo-distributed cloud data centers.

Cloud resource schedulers are typically built with a monolithic, two-level, or shared-state architecture. Monolithic schedulers, e.g., YARN, use a single, centralized scheduling algorithm for all jobs in the system. This makes them hard to scale and inflexible to support sophisticated scheduling policies. Two-level schedulers like Mesos [59] and Hadoop-on-Demand introduce a single active resource allocator to offer resources to scheduling frameworks and rely on these individual frameworks to perform fine-grained task scheduling. While being more scalable, the conservative resource visibility and locking make them hard to implement scheduling policies such as preemption and gang scheduling that require a global view of the overall resources. Shared-state schedulers such as Omega [138] aim at addressing the problems of monolithic and two-level schedulers by allowing for lock-free control. Schedulers following such designs operate completely in parallel and employ optimistic concurrency control to mediate clashes between schedulers using concepts like transactions [138], achieving both flexibility in policy customization and scalability. Adopting one of these architectures, many works have been done on the scheduling algorithm design. The heterogeneity of resources, together with the diversity of applications that impose different resource requirements, has rendered the resource scheduling problem a grand challenge for cloud computing, especially when scalability is of paramount importance.

4.1 Traditional Approaches and Limitations

Existing scheduling algorithms generally fall into one of the three categories: centralized, distributed, and hybrid. Centralized schedulers have been extensively studied, where the scheduler maintains a global view of the whole data center and applies a centralized algorithm for scheduling [13, 49, 51, 65, 161, 163, 169]. For example, Quincy [65] and Firmament [51] transform the scheduling problem into a **min-cost max-flow** (**MCMF**) problem and use existing MCMF solvers to make scheduling decisions. Considering multiple resources including CPU, memory, disk, and network, schedulers like Tetris adapt heuristics for multi-dimensional bin packing problems to scheduling. Tetrisched takes explicit constraints with jobs as input and employs a constraint solver to optimize job placement [163]. Due to the global resource visibility, centralized schedulers normally produce efficient scheduling decisions, but require special treatments to achieve high scalability.

Distributed schedulers make stateless scheduling decisions without any central coordination, aiming to achieve high scalability and low latency [126, 132]. For example, Sparrow [126] employs multiple schedulers to assign tasks to servers using a variant of the power-of-two-choices load balancing technique. Each of the servers maintains a local task queue and adopts the FIFO queuing principle to process the tasks that have been assigned to it by the schedulers. Fully distributed schedulers are known for their high scalability but may make poor decisions in many cases due to limited visibility into the overall resource usage.

Hybrid schedulers perform scheduling in a distributed manner with partial information about the global status of the data center [14, 27–29, 74]. Hawk schedules long-running jobs with a centralized scheduler while using a fully distributed scheduler for short jobs [29]. Mercury introduces a programmatic interface to enable a full spectrum of scheduling from centralized to distributed, allowing applications to conduct tradeoffs between scheduling quality and overhead [74].

Recently, a number of resource schedulers have also been proposed targeting DL workloads [103, 122, 183]. These schedulers are domain-specific, leveraging application-specific

knowledge such as early feedback, heterogeneity, and intra-job predictability to improve cluster efficiency and reduce latency.

Limitations of traditional approaches: Cluster systems are complex and often impossible to model accurately, especially in heterogeneous settings where the available resources are not uniformly distributed, or when the workload information is not known a priori. Moreover, some performance metrics like tail latency are hard to model and optimize. These properties make traditional heuristic-based solutions sub-optimal, indicating the potential benefit of learning-based solutions.

4.2 ML-based Approaches

Limited attention has been paid to applying ML techniques in general cluster resource scheduling. Paragon [30] and Quasar [31] propose heterogeneity and interference-aware scheduling, employing techniques in recommender systems, such as collaborative filtering, to match workloads to machine types while reducing performance interference. DeepRM is one of the earliest works that leverage DRL to pack tasks with multi-dimensional resource demands to servers [106]. It translates the task scheduling problem into a RL problem and employs a standard policy-gradient RL algorithm to solve it. Although it only supports static workloads and single-task jobs, DeepRM demonstrates the possibility and big potential of applying ML in cluster resource scheduling. Another attempt is on device placement optimization for TensorFlow computation graphs [118]. In particular, a RNN policy network is used to scan through all nodes for state embedding and is trained to predict the placement of operations in a computational graph, optimizing job completion time using policy gradient methods. While being effective, training the RNN is expensive when the state is large, leading to scalability issues and requiring human experts' involvement for manually grouping operations in the computational graph. In a follow-up work, a two-level hierarchical network is used where the first level is used for grouping and the second for operation placement [117]. The network is trained end-to-end, thus requiring no human experts involvement.

More recent works choose to use **directed acyclic graphs** (**DAGs**) to describe jobs and employ ML methods for scheduling DAGs. Among them, Decima proposes new representations for jobs' dependency graphs, scalable RL models, and new RL training methods [108]. Decima encodes job stages and their dependencies as DAGs and adopts a scalable network architecture as a combination of a **graph neural network** (**GNN**) and a policy network, learning workload-specific solutions. Decima also supports continuous streaming job arrivals through novel training methods. Similarly, Lachesis proposes a learning algorithm for distributed DAG scheduling over a heterogeneous set of clusters, executors, differing from each other on the computation and communication capabilities [101]. The scheduling process is divided into two phases: (1) the task selection phase where a learning algorithm, using modified graph convolutional networks, is used to select the next task and (2) the executor selection phase where a heuristic search algorithm is applied to assign the selected task to an appropriate cluster and to decide whether the task should be duplicated over multiple clusters. Such a hybrid solution has shown to provide significant performance gain compared with Decima.

ML approaches have also been attempted in GPU cluster scheduling. DL2 applies a DL technique for scheduling DL training jobs [128]. It employs an offline supervised learning mechanism is used at the beginning, with an online training mechanism at run time. The solution does not depend on explicit modeling of workloads and provides a solution for transiting from the offline to the online mechanism, whenever the latter outperforms the former.

4.3 Discussion on ML-based Approaches

While a significant amount of research has been conducted for cluster scheduling, only little focuses on applying learning techniques to fine-grained task scheduling. This could be in part explained by the complexity of modeling cluster scheduling problems in a learning framework, but also by the fact that the workloads related to the training of the scheduler need to be scheduled, possibly over the same set of resources, and that such training could be very costly. These all increase the complexity and should be included in the performance analysis and pros/cons studies of any ML-based cluster scheduling approach, which is so far widely ignored. Still, cluster scheduling can benefit from ML-based approaches, e.g., in heterogeneous settings or when the workload information is not known a priori.

5 QUERY OPTIMIZATION IN DATABASE SYSTEMS

Involving either a well-structured relational systems with SQL support or non-tabular systems (e.g., NoSQL) and in-memory stores, database systems are always the epicenter of any meaningful transaction. Yet, the state-of-the-art **database management systems** (**DBMS**), being carefully designed, remain as the performance bottleneck for plenty of applications in a broad spectrum of scenarios.

A key factor to the performance of DBMS is query optimization—determining the most efficient way to execute a given query considering all possible query plans. Over the years, several query optimizers on different levels (e.g., execution plan optimization, optimal selection of index structures) have been proposed. Most solutions rely on hand-tuned heuristics or statistical estimations. Surprisingly, case studies on query optimization reveal that the performance gains can be limited or such optimization even has a detrimental effect to the system performance, especially in the presence of estimation errors [90, 91].

5.1 Traditional Approaches and Limitations

Over the years, significant research efforts have been made on the optimization of DBMS and particularly on the query optimizer—a crucial component causally related to the query execution performance. Traditional query optimizers take as input an SQL query and generate an efficient query execution plan, or as advertised, an optimal plan. Optimizers are typically composed of sub-components, e.g., cardinality estimators and cost models, and typically involve a great deal of statistical estimations and heuristics. The main body of literature on query optimizers typically focuses on the direct optimization of a distinct component that performs better and collectively yields better results. Elaborate articles on query optimization have been published over time [18]. Nevertheless, query processing and optimization remain a continually active research domain.

An early work on query optimizers is LEO [151]. LEO gradually updates, in a process described as learning, cardinality estimates, and statistics, in turn for future use to produce optimized query execution plans. LEO utilizes a feedback loop to process history query information and adjust cost models appropriately for enhanced performance [151]. CORDS is another significant work on query optimizers, which reduces query execution time by exploring statistical dependencies between columns prior to query execution [64]. Another work Eddies supports adaptive query processing by reordering operators in a query execution plan dynamically [8]. Other contributions focus on the optimal selection of index structures [19, 53, 166]. More recent studies attempt to answer whether query optimizers have reached their peak, in terms of optimal performance, or suffer from limitations and potential performance degradation and how to mitigate them [91, 129].

Limitations of traditional approaches: The simplification assumptions based on which these query optimizers operate, do not necessarily reflect the actual data patterns underneath. As such, inaccurate estimations can lead to detrimental performance and sub-optimal execution plans. Yet, the underlying data pattern is typically too complex to capture or model by general heurisitcs or even statistical methods, rendering these traditional approaches inefficient when facing difficult data patterns.

5.2 ML-based Approaches

It is generally true that ML models are more capable of capturing a complex intuition regarding the data schemes. Thus, leveraging ML for query optimization seems a natural fit, as the exploratory nature of ML can assist in building complex estimation and cost models. Furthermore, the innate ability of ML to adapt to different environments via continuous learning can be a beneficial factor to execution plan optimizations.

5.2.1 Index Structure Optimization. Kraska et al. introduce a revolutionary approach where they investigate the overhauling of existing indexes with learned ones, based on deep learning [84]. The authors leverage a hybrid mixture of ML models to replace optimized B-trees, point indices (i.e., hash-maps), and Bloom filters. The hybrid mixture, namely **Recursive Model Index** (**RMI**), is composed of neural networks in a layered architecture and is capable of predicting index positions for B-trees and hash-maps. The simplest structure of zero hidden layers resembles linear regression, an inexpensive and fast model. Inference of the output models is done in a recursive fashion, where the top-layer neural network points to the next, and the same process is repeated until one of the base models, predicts the actual index position.

For Bloom filters, RMI does not fit. Thus, a more complex neural network with a sigmoid activation function is proposed, which approximates the probability that a key exists in the database. Comparisons with state-of-the-art approaches highlight ML as a strong rival. There are cases where reported execution speed is significantly higher, and memory footprint is sharply reduced. The novelty of the approach is a key step towards automation of data structures and optimization of DBMS, as it sets the pace for further exploration and exploitation of ML in a domain where performance is critical. ALEX extends the approach of Kraska's et al. to provide support for write operations and dynamic workflows [33].

5.2.2 Cardinality Estimation. A slightly different attempt for alleviating wrongly predicted cost models and query cardinalities is presented in [125]. The authors seek to overcome the simplifying assumptions about the underlying data structure and patterns with the respective cost estimations, which are employed through hand-tuned heuristics. To do so, a **deep neural network** (**DNN**) is utilized that learns to output the cardinality of an input query, step by step, through division into smaller sub-queries. Moving forward, the authors utilize the estimated cardinalities to produce an optimal policy via Q-learning to generate query plans. The learning agent selects query operations based on the sub-queries, which incrementally result in a complete query plan.

Kipf et al. focus on the prediction of join-crossing data correlations towards mitigating the drawbacks of current sampling-based cardinality estimation in query optimization [79]. The authors treat the problem with supervised learning by utilizing a DNN, defined as a **multi-set convolutional network** (**MSCN**), which in turn is integrated by a fully-connected multi-layer neural network. The MSCN learns to predict query cardinalities on unseen queries. Using unique samples, the model learns to generalize well to a variety of cases. More specifically, in a tough scenario with zero-tuples, where traditional sampling-based optimizers suffer, MSCN is able

to provide a better estimation. Yang et al. take an unsupervised approach to cardinality and selectivity estimation with Naru [187]. Naru utilizes deep auto-regressive models to provide high-accuracy selectivity estimators produced in an agnostic fashion, i.e., without relying on assumptions, heuristics, specific data structures, or previously executed queries.

5.2.3 Join Ordering. A more recent work called SkinnerDB targets adaptive query processing through optimization of join ordering [162]. SkinnerDB is based on a well-known RL algorithm, UTC [81]. Novelty lies on the fact that learning is done in real-time, during query execution, by slicing the query into small-time batches. It proceeds to select a near-optimal join order based on a qualitative measure, regret-bounded ratio, between anticipated execution time and time for an optimal join order [162]. SkinnerC, perhaps the most impactful variation of SkinnerDB, is able to outperform MonetDB, a specialized database engine for analytics, in the single-threaded mode, due to its ability to achieve highly reduced execution time in costly queries.

ReJOIN aims at addressing the difficulties in join order selection with a DRL approach [109]. ReJOIN formulates the RL minimization objective as the selection of the cheapest join ordering with respect to the optimizer's cost model. Each incoming query represents a discreet time-step, upon which the agent is trained on. For the task, ReJOIN employs a neural network trained on a policy gradient method. ReJOIN performs comparably or even better than the optimizer of PostgreSQL. The evaluation is conducted on a dataset specifically designed for measuring the performance of query optimizers, namely the **join order benchmark** (**JOB**) [91].

Another notable attempt is DQ [85], a DRL optimizer that exploits Q-learning in a DNN architecture to learn from a sampled cost model to select the best query plan, in terms of an optimal join sequence. To mitigate failures in cardinality estimation regarding the cost model, DQ initially converges on samples collected from the optimizer's cost model [85]. Then, weights of the neural network are stored and DQ trains again on samples gathered from real execution runs. In terms of execution, the relative Q-function is utilized to obtain the optimal join operation. Extensive evaluation indicates that DQ is remarkably more effective than ReJOIN and in a wider scope of JOB queries. Further, DQ is able to scale by incorporating more features in an effort to achieve more accurate join cost prediction.

In adaptive query processing, an early approach picks off from Eddies [8], and leverages RL to train optimal eddy routing policies [164]. The authors focus is on join and conjunctive selection queries. Additionally, the proposed framework incorporates various join operators and constraints from the state-of-the-art literature. Overall, the results indicate significance in learning an optimal query execution plan and fast reactions to changes. This work can be considered an infant step towards learned query optimizers.

5.2.4 End-to-end Query Optimization. Different from the above works that focus on distinct components of query optimizers, Neo builds an end-to-end query optimizer based on ML [110]. Neo, short for Neural Optimizer, utilizes different DL models to replace each of the components of a common optimizer. Neo relies on prior knowledge to kick-start but continues learning when new queries arrive. This approach makes Neo robust to dynamic environments, regarding unforeseen queries, albeit Neo cannot generalize to schema and data changes. Neo's contributions are manifold. Besides adaptation to changes, Neo is able to decide between three different common operations, namely join ordering, index selection, and physical operator selection [110]. Moreover, Neo integrates easily with current execution engines and users can specify their optimization objectives. Evaluations show that Neo outperforms simple optimizers and exhibits comparable performance to long-lived commercial ones.

Around the same time with Neo, SageDB conceptualizes the vision for a DBMS where crucial components, including the query optimizer, are substituted with learned ones [83]. Overall, the

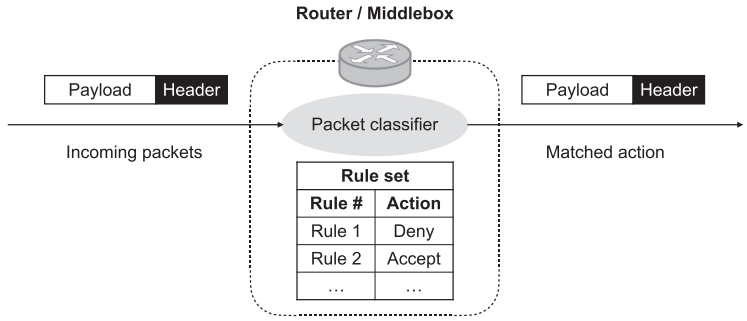

Fig. 3. Packet classifiers map packet header fields of a packet to an action following pre-defined rules.

article describes the design of such a system and how would all components tie together in a complete solution. Two approaches that concern optimization of queries in a distributed setting are Lube [172] and its sequel, Turbo [173]. Both techniques leverage ML models that concern query execution in clusters. More specifically, Lube regards minimization of response times for a query by identifying and resolving bottlenecks [172]. On the other hand, Turbo, aims at optimizing query execution plans dynamically [173].

5.3 Discussion on ML-based Approaches

Despite the decades of active research on DBMS and query optimization, it remains a fact that performance is far from optimal [90, 91, 129]. Yet, the pivotal role of databases in modern systems calls for further scrutiny. Similar to the problems in memory systems, query optimization in databases also heavily relies on the prediction of the data pattern on which ML-based approaches have demonstrated clear benefits over traditional approaches in complex scenarios. We witness the quest to yield better performance by leveraging ML in query optimization schemes. Moreover, we observe how multi-faceted those approaches are. For instance, ReJoin [109] and DQ [85] utilize DRL to tackle the selection of optimal join sequences, while Kipf et al. focus on cardinality estimation through supervised learning [79]. Additionally, when traditional approaches suffer and as modern computational units migrate to more distributed settings, we notice significantly broader approaches that target query processing accordingly (e.g., Turbo). More interestingly, we are perceiving the progress of research and how it essentially dissolves into unified schemes as recent works conceptualize end-to-end solutions (e.g., Neo and SageDB) by leveraging ML.

6 NETWORK PACKET CLASSIFICATION

Packet classification is a crucial and fundamental networking task, which enables a variety of network services. Typical examples of such network services include traffic engineering (e.g., flow scheduling and load balancing), access control, and firewall [55, 158]. A high-level overview of packet classification is depicted in Figure 3. Given a collection of rules, packet classification matches a packet to one of the given rules. Rule matching is based on certain criteria typically applied on the fields in the packet header such as source and destination IP addresses, protocol type (often including flags), and source and destination port numbers. Matching conditions include prefix-based matching, range-based matching, and exact matching. Considering the ever-increasing network traffic, packet classification dictates the need for high performance in terms of classification speed and memory efficiency. These traits need to include also a high level of classification accuracy, since mismatches can result in serious network issues such as security breaches.

6.1 Traditional Approaches and Limitations

The solution space for packet classification can be generally divided into hardware- and software-based approaches. Hardware-based approaches typically leverage **ternary content addressable memories** (**TCAMs**) and are considered the standard in industrial high-performance routers and middleboxes. TCAM is a specialized type of high-speed memory, which stores matching rules as a massive array of fixed-width entries [89] and is able to perform multi-rule matching in constant time. Early work also extends TCAMs to increase performance on lookups and reduce power consumption by utilizing a special storage block that is indexed before resolving to subsequent lookups [146]. Although the use of TCAMs significantly boosts classification speed, these solutions have inherit limitations including poor scalability (e.g., in-range expansion), high cost, and high power consumption [89].

On the other hand, software-based approaches offer greater scalability but suffer performance-wise in general. A representative family of software-based approaches is based on tuple space introduced in [148]. These approaches partition rules into tuple categories and leverage hashing keys for accessing the tuple space of a particular filter [158]. While yielding fast queries, the hashing induces non-deterministic speeds on look-ups or updates [54]. Another family of software-based approaches is based on decomposition. A noteworthy work in this family is DCFL [159], which takes a distributed approach to filter searching. In particular, independent search engines match the filter fields and aggregate the results in an arbitrary order [158]. However, this technique mandates multiple table accesses, thus impacting performance [167].

Most software-based packet classification approaches are based on decision trees. The idea is to classify packets by traversing a set of pre-built decision trees and selecting the highest priority rule among all matched rules in one or more decision trees. To reduce the classification time and memory footprint, decision trees are optimized to have small depths and sizes based on hand-tuned heuristics like node cutting or rule partitioning [54, 142]. EffiCuts, which builds on its predecessors HiCuts [54] and HyperCuts [142], significantly reduces memory footprint by employing four heuristics: separable trees, selective tree merging, equi-dense cuts, and node co-location [167]. A more recent work, CutSplit, optimizes decision trees on the premises of reducing rule overlapping, unoptimized yet faster first stage cuttings, and by effective pre-cutting and post-splitting actions [94]. Another work leverages decision trees and TCAMs in a hybrid approach [82].

Limitations of traditional approaches: Current hardware- and software-based solutions pose strong limitations to effective packet classification. As discussed, hardware approaches fall short in terms of scalability and exhibit significant monetary and power costs, while software solutions rely on hand-tuned heuristics. Heuristics can be either too general to exploit the characteristics of a given rule set or too specific to achieve good performance on other rule sets. In addition, the lack of a specific, global optimization objective in the heuristic design can result in sub-optimal performance. Finally, the incorporation of different heuristics into a single solution can incrementally increase the overall complexity of the approach, hindering optimization due to difficulty in understanding them.

6.2 ML-based Approaches

ML for packet classification typically replaces the classifier with a model pre-trained with supervised learning. However, with the recent advances in DRL, the solution space of packet classification approaches broadens. In general, there are three categories: (1) using supervised learning to replace the classifier with a trained model, (2) using RL agents to generate suitable decision trees

at runtime according to the given set of rules, and (3) leveraging unsupervised learning to cluster unforeseen traffic.

Approach (1) fits naturally since packet classification is by definition a classification task. These approaches commonly utilize a traditional supervised learning setting where information concerning incoming traffic is known a priori and traffic is classified into distinct labeled sets. Traditional supervised learning proposals for packet classification have also been reviewed extensively in [124]. Yet, the first remarkable work in this direction that leverages DL targeting traffic classification is only recently introduced with Deep Packet [100]. This work leverages **convolutional neural networks** (**CNNs**) to construct a traffic classifier that is able to characterize traffic and identify applications without given advanced intelligence (i.e., hand-tuned features). Despite the promising results, the accuracy requirement of packet classification renders the neural network-based approach impractical. This is because neural networks cannot guarantee the correct matching of rules. Moreover, the size of the neural network has to be big enough in order to handle a large set of rules. Thus, achieving high performance is very unlikely without hardware accelerators like GPUs [97]. Further, supervised learning schemes are generally limited by design as supervised learning necessitates certain information is known in advance [130].

Approach (2) learns at the meta-level where we learn to generate appropriate decision trees and use the resulting decision trees for actual packet classification. This way, ML methods are out of the critical path so performance is no longer an issue. NeuroCuts is to the best of our knowledge the first work that employs a DRL method for decision tree generation [97]. NeuroCuts employs a DRL approach in a multi-agent learning setting by utilizing an actor-critic algorithm based on **Proximal Policy Optimization** (**PPO**) [137]. An agent executes an action in each discrete time step with the target of obtaining a maximized reward. The action depends on the observed environment state. Following the same footsteps, Jamil et al. introduce a classification engine that leverages DRL to generate an optimized decision tree [70]. In detail, the derived tree concentrates the essential bits for rule classification into a compact structure that can be traversed in a single memory access [70]. Then, the outcome of the traversal of the generated tree is utilized in the original tree to classify packets. This results in a lower memory footprint and higher packet classification speed.

Finally, following approach (3), Qin et al. leverage an unsupervised learning scheme to mitigate drawbacks of prior supervised learning solutions [130]. As mentioned in their work, existing supervised approaches fail to adjust to network changes as unforeseen traffic arrives and classification performance deteriorates. Besides, the authors advocate in favor of link patterns as a crucial property on network knowledge, while most approaches utilize only packet-related features. They propose a novel combinatorial model that considers both sources of information (packet and link patterns), in a clustering setting [130]. The approach is evaluated against several baselines of supervised and clustering algorithms and is able to outperform all of them, building a strong case for traffic classification with unsupervised learning.

6.3 Discussion on ML-based Approaches

Recent works in packet classification provide us with several useful insights. First, we observe that technological advancements in ML drive stimuli in the way researchers approach now the challenge of packet classification. For instance, typical solutions that used to solely focus on training packet classifiers have now diverged to more radical, unorthodox approaches. Second, we can see that significant effort has been put towards leveraging DRL-based solutions, perhaps the most recently advanced and trending research domain for the past few years. Third, we observe that ML often entails more performance metrics than common approaches. For example, classification accuracy is a critical metric in packet classification with supervised learning, whereas hardware-based

approaches (e.g., TCAMs) do not impose such constraints. Finally, we can deduce that as the scope of work widens, more and more works that target similar directions will be explored and proposed. For example, Li et al. propose a novel way of caching rules into memory with LSTM neural networks, which can be directly exploited for packet classification [93]. Overall, ML-based approaches address the limitations of traditional approaches by being more generalizable, being able to incorporate complex optimization goals, and reducing the design complexity. However, they still fall short for critical scenarios due to the lack of guarantee in results and explainability. In scenarios where accuracy is of critical importance, traditional approaches would still be preferable.

7 NETWORK ROUTING

Traffic Engineering (**TE**) is the process of optimizing performance in traffic delivery [175]. Perhaps the most fundamental task of TE is routing optimization, a path selection process that takes place between or across networks. More specifically, packet routing concerns the selection of a path from a source to a destination node through neighboring nodes. In each traversing node, routing aims at answering the question of which adjacent node is the optimal node to send the packet. Common objectives of packet routing involve optimal time to reach the destination, maximization of throughput, and minimum packet loss. It should be applicable to a broad variety of network topologies.

7.1 Traditional Approaches and Limitations

Routing is a broad research subject with a plethora of differing solutions and approaches proposed over the years. Routing commonly differentiates between intra- and inter-domain. The former concerns packets being sent over the same **autonomous system** (**AS**) in contrast with the latter, which regards sending packets between ASes. Routing can also be classified based on enforcement mechanisms or whether it concerns offline/online schemes, and furthermore on the type of traffic per se [175]. Based on this wide taxonomy, which only expands with emergent network topologies, there are distinct types of proposed solutions, as well as research that typically considers a more fine-grained domain of routing optimization. As our interests lies mostly in computer systems, we concentrate mainly on intra-domain traffic engineering. A comprehensive survey that covers routing optimization in a coarse-grained manner that concerns traditional networks and topologies is presented in [175]. For surveys on routing in wireless sensor networks and ad-hoc mobile networks, we refer the readers to [5].

Regarding intra-domain traffic engineering, **open shortest path first** (**OSPF**) solutions are prevalent favoring simplicity but often suffer in performance. OSPF solutions cope well with scalability as network growth has reached an all-time high, but have pitfalls when it comes to resources utilization [115]. As in the case of packet classification, common OSPF proposals are based on hand-tuned heuristics [147]. Moreover, most of existing literature that aims at meditating the performance boundaries set by OSPF approaches, have seen rare actual implementation [115].

In addition, we find it necessary to add some notes about **software-defining networking** (**SDN**). Conventional, non-SDN, network devices embed dedicated software to implement networking logic, such as packet routing, and are characterized by long and costly evolution cycles. SDN reduces networking devices to programmable flow-forwarding machines, with networking logic now running at a logically centralized controller, adding more flexibility in programming the networking behaviors [112]. SDN can therefore be seen as a way to implement routing decisions at network devices, but it does not change the nature of the routing problem, which now needs to be solved by the controller. We do not go further into such implementation details and refer the readers to published surveys in this area, e.g., [42].

Limitations of traditional approaches: The main challenge of network routing consists in the ever-increasing dynamics of the networks, including the traffic loads as well as the network characteristics (e.g., topology, throughput, latency, and reliability), and the multi-faceted optimization goals (reflecting the user quality of experience ultimately), which are hard to be interpreted as a simple formula for handcrafted heuristics to optimize. The implementation complexity of network routing optimization is also a practical concern.

7.2 ML-based Approaches

The first work involving ML on the challenge of traffic engineering dates back to 1994, namely Q-Routing [15]. Leveraging a fundamental RL algorithm, Boyan et al. propose a learning-based approach to tackle the problem. Q-Routing derives from Q-learning [177], and is able to generate an efficient policy with a minimization objective—the total time to deliver a packet from source to destination. By conducting a series of experiments on different network topologies and dynamic networks, Q-Routing exhibits significant performance gains, especially on congested network links, over static approaches. More importantly, Q-Routing establishes RL as a natural fit to the problem and paved the way for research on learnt systems for traffic engineering.

Since the introduction of Q-Routing, several works have emerged that utilize alternative RL methods or study various network topologies and scope of applications. A comprehensive survey of these solutions w.r.t. all known network topologies, i.e., from static to dynamic vehicular and ad-hoc networks, is provided in [104]. However, it is clear from the survey that when it comes to implementation, the tremendous state and action space quickly becomes a hefty burden in learning and thus results in sub-optimal solutions. The application of DL to network traffic control and essentially routing optimization has been studied in [44, 105, 165]. In detail, Fadlullah et al. take a high-level overview of what is comprised as a supervised learning scheme with **deep belief networks** (**DBNs**) [26], which predicts on a node basis the next path (i.e., router) to deliver to and in turn, the destination node does the same and so forth [44]. In this approach, each node is solely responsible for outputting the next node and the full delivery path is uncovered in a hop-by-hop manner. Interestingly enough, Mao et al. take a similar supervised approach with **Deep Belief Architectures** (**DBAs**) in their work [165]. Even though both approaches are novel and interesting first-steps, they are applied in a constrained static setting raising questions that come naturally as how supervised learning can be scaled and applied efficiently into dynamic and large network topologies.

Mao et al. bring RL back on the table in [165]. Initially, the authors evaluate a supervised learning scheme through varying DNN architectures. By observing past **Demand Matrices** (**DMs**) the neural network learns to predict the DM which is then leveraged to calculate the optimal routing strategy for the next epoch. The evaluation results, however, show that supervised learning is not a suitable approach for dynamic settings. The authors therefore employed DRL and interchanged the prediction of DMs to learn a good mapping policy. Also, the design shifts to a more constrained setting, focusing on destination-based routing strategies. The agent's reward is now based on max-link-utilization, and the algorithm of choice is **Trust Region Policy Optimization** (**TRPO**) [136]. As the large action space can cripple the learning process, the number of output parameters is reduced by shifting learning to per-edge weights. While this DRL-based mechanism yields better results, compared with the supervised solution, the proposed solution is not significant enough to alter the domain of routing as it is.

To mitigate the risk of state space explosion and significant overhead of globally updating a single agent, a distributed, multi-agent approach is applied in [191]. In particular, You et al. pick up

where [15] left off. As a first step, they upgrade the original Q-Routing contribution by exchanging the Q-Table with a DNN, namely **deep Q-Routing** (**DQR**), but leaving the remainder of the algorithm pristine. In contrast to the semantics of the approach, the authors differentiate their proposal by specifying a multi-agent learning environment. As such, every network node holds its own agent, and each agent is able to make local decisions deriving from an individual routing policy.

Limitations of the proposed ML and DL schemes to revolutionize the domain of routing led Varela et al. to argue that simply applying state-of-the-art algorithms and techniques is not sufficient when it comes to networking challenges [153]. Instead, Varela et al. shift their focus on feature engineering and further outline that a complete yet simple state representation might be the key to overcome the hurdles [153]. Furthermore, Varela et al. propose a DRL scheme that integrates telemetry information alongside path level statistics to provide a more accurate representation for the purpose of learning [153]. Reportedly, their proposed scheme integrates better to various network configurations.

Other recent works also base their ideas on feature engineering and argue that achieving generalization is the key to success [99, 134], especially when dealing with network dynamics. Rusek et al. take a supervised learning approach with GNNs introducing RouteNet that aims at generalizing over any given network topology by making meaningful predictions on performance metrics [134]. Using these predictions, it is able to select a suitable routing scheme that abides by the environment constraints. Meanwhile, DRL-R contributes a novel combinatorial model that integrates several networks metrics to address shortcomings of existing DRL schemes [99].

7.3 Discussion on ML-based Approaches

While a lot of effort has been made on the fundamental challenge of routing optimization, we are confident to say that the task is far from complete and continues to remain an active research domain. Purely from an ML perspective, we can obtain several useful insights for aspiring scholars and researchers. For one, while routing optimization fits naturally to RL-based approaches, this does not strictly bind that achieving optimal results is a matter of learning paradigm. We saw several works attempting to leverage supervised learning techniques for that matter. On the other hand, taking into consideration that most DL approaches initiated with supervised learning and then shifted to RL, evidence might suggest otherwise. Besides, we observe the great potential of applying distributed, multi-agent, DRL-based approaches, as they can effectively mitigate the risk of state and action space explosion and improve the generalization properties of the learned algorithms. These traits make them a better fit to address the routing problems in large and dynamic environments such as carrier networks.

8 CONGESTION CONTROL

Congestion Control can be characterized as a remedy for crowded networks [66]. It concerns actions that occur in event of network changes as a response to avoid collisions and thus network collapse. Network changes in this domain regularly refer to changes in the traffic pattern or configuration, resulting in packet losses. A typical action to avoid collapse is for the sender to decrease its sending rate, e.g., through decreasing its congestion window. TCP, the de facto network transport protocol that the Internet relies on for decades, suffers from many limitations. With TCP being architecturally designed at 1980s, it is natural that the original specification contains network behaviors observed at the time [178]. It is also the case that emerging ad-hoc and wireless networks are being hampered by the lack of flexibility inherited by TCP [9]. That being said, it has been shown that the congestion control scheme of TCP is often the root cause for degraded performance. Interesting literature has displayed the symptoms of current congestion control mechanisms, such as bufferbloat [48] and data-center incast [20].

8.1 Traditional Approaches and Limitations

Congestion control has drawn a lot of research attention in the past decades and is still a very active domain. Various techniques have been proposed, mainly relying on human expert designed heuristics. In particular, IETF has proposed a series of guidelines to aid network designers in meditating TCP's innate drawbacks [75]. These mechanisms usually apply end-to-end techniques, altering TCP by tuning the congestion window rolling as a mean to achieve better performance. This is done based on a number of factors and often simplifying or constraining assumptions about network conditions. Placing significant approaches in a chronological order we display quite a long list of literature: Vegas [16], NewReno [61], FAST TCP [157], CUBIC [56], and BBR [17]. Other more recent approaches with a focus on subsets of congestion control such as short-flows or datacenters are found in [92, 119, 180, 193]. Extension of congestion control to multipath scenarios, i.e., multipath TCP [46], have been explored in various studies, notably [77] and [127]. As it is not feasible to cite all the related work, we refer the readers to [88] for further discussion about congestion control of TCP and its many different variants.

Limitations of traditional approaches: The assumptions commonly formulated regarding network conditions in the aforementioned literature are often far from being realistic. Despite the evaluations demonstrating significant results, applications of traditional solutions to real-network traffic exhibit strong signs of degraded performance and underutilization of resources [34]. This is where applying ML techniques, and specially model-free approaches such as DL, become useful.

8.2 ML-based Approaches

In the congestion control scenario, ML is capable of setting clear and direct optimization objectives to eliminate the rather unknown goals that the current setting holds. Additionally, with ML we can generate online learning control algorithms that are able to adjust to constantly changing network conditions. To incrementally progress towards our goal, existing domain knowledge can be incorporated to arrive at better learning solutions.

To the best of our knowledge, Remy [179] is the first to utilize ML for learning a congestion control algorithm. Remy formulates an offline learning environment, in a decentralized manner, similar to what we have seen already in Section 7.2 as POMDPs. Agents sit on the endpoints and every time step takes a decision between send and abstain. Remy is trained under millions of sampled network configurations, through which it is able to come along with the optimal control strategy within a few hours of training. RemyCC, the output control algorithm, is employed on the current TCP stack and the evaluation suggest that it is able to outperform several state-of-the-art solutions. However, Remy does not manage to escape the pitfall of underlying assumptions. The training samples that Remy builds upon place constraints on RemyCC due to assumed network configurations under which they are sampled. This limits Remy's state-action space, and can heavily affect performance when those conditions are not met.

In contrast to Remy, PCC Allegro [34] attempts to tackle inherit limitations of predefined assumptions on the network by conducting micro-experiments. In each experiment, an appropriate control action is taken based on which the learner optimizes towards a "high throughput, low loss" objective named utility. Then, Allegro learns empirically to make better decisions by adjusting to control actions that yield higher utility. Following Remy's example, each sender in Allegro makes decisions locally based on the outcome of the micro-experiments [34]. Allegro scheme does not make assumptions about network configurations. This translates to sharply

greater performance in real-network scenarios. Despite the effort, Allegro's convergence speed and towards-TCP-aggressiveness frame it as prohibitive for deployment [35].

In an attempt to eliminate Allegro's limitations, Dong et al. introduce Vivace [35]. Vivace replaces the two key components of Allegro: (1) the utility function and (2) the learning algorithm of rate-control. For the utility function, Vivace integrates RTT estimations via linear regression to penalize for high latency and loss [35]. Through latent-aware utility, Dong et al. show that Vivace can achieve fairness while mitigating bufferbloat and remaining TCP-friendly. For the rate-control algorithm, Vivace employs gradient-ascent-based no-regret online optimization [199]. The no-regret part translates to a minimum guarantee towards performance. Further, Vivace's rate-control scheme enables faster convergence and subsequently faster reaction to network changes [35].

Pantheon was initiated as a playground with a focus on congestion control, where researchers can benchmark their proposals with other state-of-the-art literature and evaluate performance on shared metrics and measurements [186]. Pantheon leverages transparency by keeping a public record of results. Besides the shared platform for knowledge, the authors introduced Indigo—an offline neural network-based approach to congestion control. Indigo utilizes an LSTM RNN [60] and trains it with an imitation learning algorithm called DAgger [133]. Indigo employs generated data from optimal solutions that display correct mappings from state to action. Based on this training, Indigo is able to adjust its congestion window once an ACK is received [186]. Indigo exhibits relatively comparable performance to other schemes in Pantheon's platform.

Aurora employs DRL to extend Allegro and Vivace [71]. Similarly to these solutions, it controls the sending rate per time-step, but the learning scheme incorporates into its observed history both latency gradient and ratio from [34, 35] respectively, as well as sending ratio which is specified as the ratio of packets sent to packets acknowledged by the receiver. The reward setting praises throughput and penalizes latency and packet loss, with packet loss referring to packets that have not been acknowledged. A RL agent is trained with a utilization of an algorithm first introduced in Section 6.2, namely PPO. With a relatively simple neural network configuration, Aurora is able to generalize well to a good mapping of sending rates outside of its environment scope. This establishes Aurora as a robust solution which can be applied to dynamic networks which exhibit unpredictable traffic conditions. In contrary to its robustness, Aurora is comparably similar in terms of performance to other state-of-the-art schemes.

Most recent studies tackle issues related to generalization and convergence speed. Specifically, a practical, hybrid approach is proposed in [1], which combines classic congestion control techniques with DRL techniques, improving the generalization toward unseen scenarios. The idea is to have two levels of controls: fine-grained control using classic TCP algorithms, e.g., BBR, to adjust the congestion window, and hence the sending rate, of a user, and coarse-grain control using DRL to calculate and enforce a new congestion window periodically, observing environment statistics. The proposed solution, therefore, has more predictable performance and better convergence properties, showing how learning from an expert, e.g., BBR algorithm, can improve the performance, in terms of convergence speed, adaptation to newly seen network conditions, and average throughput [41].

Applying DRL to multipath scenarios is also getting a boost. Notably, a centralized solution is proposed in [184], with a single agent trained to perform congestion control for all the MPTCP flows in the network. Such centralized solutions, however, are not scalable, as they require a global view of all available resources and active MPTCP flows in the network. A distributed solution is proposed in [95], where multiple MPTCP agents, each running at a sender node, are learning a set of congestion rules that enable them to take appropriate actions observing the environment. The learning is performed in an asynchronous manner, where each node requires only local environment, state, and information. Further, as state is defined in a continues, high-dimensional space,

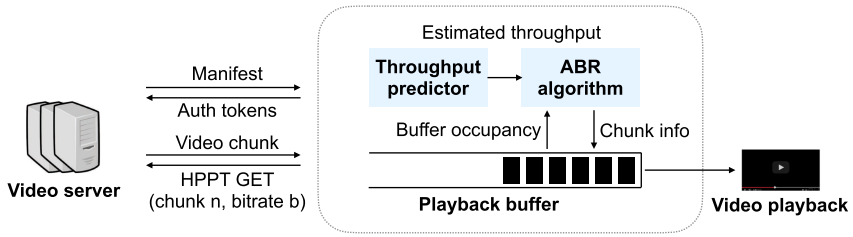

Fig. 4. An overview of adaptive video streaming.

tile coding methods are applied to discretize the state dimension, addressing the scalability issue. The proposed solution, however, relies on offline learning and hence has limited generalization capabilities. In contrast, an online convex optimization is explored in [50], which extends PCC to multipath settings, showing through theoretical analysis and experimental evaluation that the proposed online-learning solution is scalable and that it can significantly outperform traditional solutions and better adjust to the changes in the network conditions. However, no comparison among these DRL methods is provided.

Finally, DeePCCI [135] proposes a novel classification scheme for identification of congestion control variants using DL. The authors solely-regard the packet arrival time of a flow as their input arguing on the fact that congestion control is strongly associated with packet timings. Additionally, fewer features directly translate to the ease of adaptation and applicability to other congestion control schemes. The classifier is trained and evaluated against CUBIC, RENO, and BBR with generated labeled data as necessitated for supervised learning.

8.3 Discussion on ML-based Approaches

As one of the oldest and most established areas in networking, congestion control has attracted rich research attention, in both traditional and ML-based domains, with many articles published recently to adopt DL techniques. These studies show the capability of DL techniques, and specifically DRL techniques, to overcome the limitations of traditional solutions. By learning adaptive mechanisms, these ML-based solutions are able to adjust to constantly changing network conditions and hence better utilize the resources. The majority of these solutions, however, focus on centralized or off-line learning, with only very few tackling online learning in a distributed setting—the case for TCP. More work should be done in this direction. Besides, most of these studies focus on proposing solutions that outperform current mechanisms, without really changing the objectives and the goals.

9 ADAPTIVE VIDEO STREAMING

As multimedia services such as video-on-demand and live streaming have witnessed tremendous growth in the past two decades, video delivery, nowadays, holds a dominant percentage of overall network traffic on the Internet [170]. Current video streaming services are mostly based on ABR, which splits a video into small chunks (a few seconds long) that are pre-encoded with various bitrates and streams each of the chunks with a suitable bitrate based on the real-time network condition. ABR is behind many mainstream HTTP-based video streaming protocols including **Microsoft Smooth Streaming** (**MSS**), Apple's **HTTP Live Streaming** (**HLS**), and more recently **Dynamic Adaptive Streaming over HTTP** (**DASH**) standardized by MPEG [143].

An overview of adaptive video streaming is depicted in Figure 4. The bitrate selection for each chunk is dictated by an ABR algorithm running on the client side, which takes real-time

throughput estimations and/or local buffer occupancy as input and optimizes for the quality of experience defined as a combination of metrics including re-buffering ratio (the percentage of time the video playback is stalled because of drained buffer), average bitrate, bitrate variability (to improve playback smoothness), and sometimes also the startup delay (the time spent between user clicking and the playback starts). Considering that the network status suffers from high dynamics, designing a good ABR algorithm is non-trivial. This has been confirmed in an early measurement study which shows significant inefficiencies of commercial and open-source ABR algorithms [3]. As a result, many new ideas have been explored to improve adaptive video streaming. Here, we focus on the advancements on client-side ABR algorithms.

9.1 Traditional Approaches and Limitations

Early ABR algorithms can be generally categorized into two families: rate-based and buffer-based (including the hybrid ones). Rate-based ABR algorithms typically rely on estimating network throughput based on past chunk downloads information [72, 96, 171, 200]. The ABR algorithm then selects the highest possible bitrate that can be supported by the predicted network throughput. To reduce prediction variability, ABR algorithms usually smooth out the predictions. For example, FESTIVE uses the experienced throughputs of the past 20 samples to predict the throughput for the next chunk and adopts the harmonic mean to reduce the bias of outliers [72]. Noticing that the measured TCP throughput may not reflect precisely the available real network throughput, PANDA proposes a "probe and adapt" method, similar to TCP's congestion control, but at the chunk granularity to stress test the real network throughput [96]. SQUAD takes running estimates for the network throughput which acknowledges the impact of the underlying TCP control loop and takes into account the time scale to improve smoothness and reliability. It then uses a spectrum-based adaptation algorithm for the bitrate selection [171].

Buffer-based ABR algorithms leverage the buffer occupancy as an implicit feedback signal for bitrate adaptation [62, 160], often also in combination with throughout prediction [189]. Huang et al. advocate for a pure-buffer-based design, which incorporates bandwidth estimation whenever needed (e.g., during the session startup). They model the dynamic relationship between the buffer accuracy and bitrate selection and propose a buffer-based ABR algorithm called BBA [62]. Similarly, BOLA is solely buffer-based, where the bitrate adaptation problem is formulated as a utility maximization problem and solved by an online control algorithm based on Lyapunov optimization techniques [145]. **Model predictive control** (**MPC**) is a hybrid approach that integrates both the throughput and the buffer occupancy signals [50]. More specifically, MPC models bitrate selection as a stochastic optimal control problem with a moving look-ahead horizon and leverages MPC to perform bitrate selection. To reduce the high computation, FastMPC proposes to use a table enumeration approach instead of solving a complex optimization problem as in MPC. ABMA+ pre-computes a buffer map which defines the capacity of the playout buffer required under a given segment download condition to meet a predefined rebuffering threshold and uses the map to make bitrate adaptation decisions [10].

Limitations of traditional approaches: ABR algorithms typically rely on accurate bandwidth estimation which is hard to achieve with simple heuristics or statistical methods. Also, ARB algorithms make adaptation decisions based on the bandwidth estimation with a complex relationship between the two, rendering simple heuristic approaches general to all scenarios ineffective.

9.2 ML-based Approaches

As discussed above, ABR algorithms typically involve bandwidth estimation, a complex control logic, or both, where we can leverage existing ML methods: the bandwidth estimation problem can be treated as a general regression problem, while the control problem can be treated as a decision-making problem. In fact, existing work on applying ML in adaptive video streaming can be generally divided into these two lines. We will discuss these two lines separately.

The first research line aims at achieving better accuracy in bandwidth estimation. Bandwidth estimation is a general and well-studied problem that has its presence in many of the Internet applications [68, 69]. Existing ML-based approaches for bandwidth estimation mainly focus on using methods like Kalman filter [40] or neural networks [43, 78] to perform the prediction. In the context of adaptive video streaming, CS2P aims at improving bitrate adaptation by adopting data-driven approaches in throughput prediction [154]. The authors make two important observations: (1) There are similarities in the throughput pattern across video streaming sessions. (2) The throughput variability within a video streaming session exhibits stateful nature. Based on the first observation, the authors cluster similar sessions and use the clustering result to predict the initial throughput of a session. Using the second observation, they propose a Hidden-Markov-Model-based method to explore the stateful nature of throughput variability to predict the throughput.

The second line of research focuses on leveraging ML, RL in particular, techniques for adaptation decision making. By penalizing on detrimental factors that negatively affect the optimization objectives, RL can learn from expererience an optimal strategy for bitrate selection, and replace existing heuristic-based schemes that suffer in generalizing to dynamic networks. RL can exploit the low level system signals that are essential for ABR algorithm design but are hard to be modeled and considered due to the inherent complexity. Besides, RL-based approaches are typically more flexible and can be generalized to different network conditions.

Claeys et al. try to replace used heuristics in the HTTP adaptive streaming client by an adaptive Q-learning-based algorithm [25]. Q-learning is a model-free RL algorithm that can make bitrate adaptation decisions by calculating the Q-value representing the quality (i.e., QoE) of decisions under varying network conditions. While showing promising results, this initial attempt is bounded by an explosive state-action space that burdens the convergence speed, resulting in slow responses to network variations. In a follow-up work [24], the authors apply several optimizations using a variant of **Frequency Adjusted** (**FA**) Q-learning. The new method alters Q-value calculation leading to quicker fitting in network fluctuations. In addition, the authors significantly reduce the environment state parameters and manage to achieve faster convergence.

Several studies formulate the bitrate adaptation problem as a **Markov Decision Process** (**MDP**) with a long-term reward defined as a combination of video quality, quality fluctuations, and rebuffering events [21, 47, 107, 198]. As one example, mDASH proposes a greedy algorithm to solve the MDP, which results in suboptimal adaptation decisions but is efficient and lightweight [198]. Chiariotti et al. [21] propose a learning-based approach leveraging RL to solve the MDP. To boost learning speed, the proposed learning approach utilizes **Post-Decision States** (**PDSs**) [111]. In combination with what is known as off-policy learning and softmax policy, the proposed approach is able to converge fast enough to react to network changes.

To combat the innate limitations of Q-learning, D-DASH leverage DL where instead of enumerating all the Q-values a DNN is used to approximate the Q-values [47]. Tradingoff performance with converge speed, D-Dash employed four variations of DNN architectures, and tested under several environments. Comparable to the state-of-the-art and fairly fast, D-Dash definitely sets the pace for more DL approaches in upcoming research. Another DL-based approach is called Pensieve [107], which differentiates from the herd by utilizing a DNN that is trained with state-of-the-art

A3C algorithm [120]. Pensieve is trained offline in a multi-agent setting that speeds up learning on a plethora of network traces. Besides evaluations in a simulated environment, Pensieve is also tested against real network conditions. Pensieve is able to outperform baseline ABR algorithms on shared QoE metrics. Moreover, Pensieve's ability to incorporate throughput history is a key aspect.

The above list of ABR algorithms is far from complete. For a comprehensive survey of existing ABR algorithms please refer to [11]. Apart from designing new ABR algorithms, researchers have also explored how to tune a given ABR algorithm under various network conditions. For example, Oboe is such a method that auto-tunes ABR algorithms according to the stationarity of the network throughput [4]. In [168] the authors propose to leverage RL to configure the parameters in existing ABR algorithms, achieving better bandwidth awareness.

9.3 Discussion on ML-based Approaches

Adaptive video streaming is a well-formulated problem that has attracted tremendous research efforts in the past decade. Both traditional approaches based on heuristics or control theory and ML-based approaches have been explored. It is evident that DRL-based approaches have a great potential for bitrate adaptation due to its innate advantage of capturing complex variability in network bandwidth [107]. Choosing the right data set and allowing for enough training seem critical to the performance of DRL-based approaches, and it is unclear if solutions like Pensieve can be generalized to any network environments with varying conditions. Overall, DRL-based approaches are more capable of capturing the network dynamics and the complex relationship between bandwidth estimation and adaptation decisions than traditional heuristics-based approaches. Yet, DRL-based approaches require a lot of data and resources to train an accurate model, which can be an intimidating factor for many video streaming service providers. In such cases, traditional approaches based on control theory might be a better option.

10 DISCUSSION AND FUTURE DIRECTIONS

Despite all the praise and recent remarkable publications that regard the application of ML in computer systems, ML is not panacea and should not be treated as such. There are still challenges that lie ahead before deploying learned systems in real-world scenarios. So far, we have witnessed mostly the beneficial progress of reviewed applications and albeit this is sufficient to intrigue researchers to investigate more, we also ought to raise awareness when it comes to integration of ML techniques in complex systems. That being said, in this section, we aim at outlining and discussing current known limitations in the literature we have reviewed, whilst, offering approaches that might assist in future research.

Explainability. Some recent studies [32, 197] try to shed light on the limitations of ML systems. Interestingly, both address ML-based solutions as "black-box approaches" and focus on the lack of interpretability of ML models. Especially in DNNs with multiple hidden layers, we cannot sufficiently capture, or better yet rationalize, the logic behind the decision-making process of these complex models and architectures. As further discussed in [197], this leads to several ambiguities and trust issues, especially when it comes to the process of debugging such complex structures. On the other hand, simpler and intuitive models are burdened with deteriorated performance and poor generalization [32]. Moreover, DNNs are subjective to unreliable predictions when input does not match the expectations assumed on training [197]. Specific research questions to explore include: (1) How to come up with clear guidelines (e.g., for determining DNN architectures and hyper-parameters) on the design of ML-based solutions? (2) How to open up the black box of DNNs and incorporate domain expertise in the decision-making process?

Training overhead. Besides explainability, training times and the associated costs are another significant drawback. For instance, as we have seen in [188], where each video is trained separately

for efficient delivery, it requires approximately ten minutes of training per minute of video or in cost-wise, 0.23 dollar. Putting it into perspective, consider the well-known YouTube platform. YouTube, the de facto platform for video streaming is estimated to have around 500 hours of videos uploaded every minute [150]. In combination with training time, it would be impossible to handle the training for all videos with such tremendous growth. In terms of cost efficiency, consider that only a relatively small amount of videos from the total uploads will convert into revenue, hence, we can deduct that supporting such a large-scale content-aware delivery system is practically impossible. The following research questions would be interesting to explore: (1) Can we train the DNNs with just a small amount of data? (2) How to apply transfer learning to reuse DNNs in scenarios that are similar in structure but different in detail?

Lack of training data. Related to training times, lies also the concern of training data. Due to the fact that sufficient and effective training necessitates large volumes of data, many of the existing studies rely on generated samples or existing datasets to train their models. For instance, both Wrangler [185] and CODA [196] train on datasets composed of traces that stem from production-level clusters. On the other hand, MSCN [79] generates training data from sampled queries that are based on "schema and data information". This results into two major concerns. First, it has to be ensured that training samples are sufficient and cover the whole problem space [76]. Otherwise, we might be looking into a solution that does not generalize well in real-world scenarios. Second, as mentioned in [197], there are adversarial inputs that can be the root cause of degraded performance. The following research questions need to be answered: (1) Can we build a common training ground for DNN training without leaking sensitive data to the public? (2) How to achieve scalable incremental training so the system can learn on the fly over time?

Energy efficiency. Training a large DNN could also has substantial environmental impacts. A recent study [152] shows that the carbon emission of training a BERT model on current GPUs is roughly equivalent to a trans-American flight. To reduce the carbon emission of such training, one can improve the hardware design or the algorithm, to reduce the complexity or training time. Harvesting green energy resources, such as solar cells or wind turbines, for training is another way to make such training greener and hence more sustainable. But this requires advances in distributed learning technologies, where the training of a large DNN can be distributed all over the network, and closer to the edges where green energy resources are available. Despite some advances in the field of distributed learning, e.g., by proposing FL [113], the green learning aspect of it has not yet been covered. We identify the following specific research questions: (1) How to design more energy-efficient ML methods (for both training and inference)? (2) How to use distributed learning techniques to avoid bulky energy consumption in central places?

Real-time performance. A final implication of ML approaches is that of inference time. Real-time applications require fast inference of ML models, as in the case of autonomous driving, where hardware accelerators are employed so the system can make safety-critical decisions locally [98]. This hurdle composes also the main idea and contribution of [176], where supervised learning is applied to enhance vision analytics tasks on cloud operations. Expanding to other domains where speed is crucial, e.g., routing and congestion control, we can understand how slow inference of models, especially in DNNs, can reduce the performance gains significantly. This can add up to an overall comparable performance with traditional solutions, in which an operator will prefer the traditional solution for simplicity and better understanding [197]. Specific research questions include: (1) How to make ML-based methods lightweight when applied in the critical path of computer systems and networks? (2) Can we use a lightweight method on the critical path while leveraging ML-based methods on the non-critical path?

Taking all aforementioned limitations into account, it is safe to suggest that ML for computer systems and networking is still at an infant stage. Even though we have witnessed remarkable

progress, we need to tread lightly when it comes to deploying systems that heavily depend their operation on ML approaches, due to the fact that we might end up with a sub-optimal solution, compared to the one we are seeking to alleviate. While it is unclear how to integrate existing domain knowledge [32], it is argued that this is the key to overcome current drawbacks [197]. Doing so, the authors suggest that commons limitations, and in particular transferability, robustness and training overhead will be essentially mitigated. Furthermore, a recent work from Kazak et al. proposes a novel system to verify DRL systems [76]. Verily, the aforementioned system is a first step towards formal verification of DRL models. The author's contribution aims at verifying that learned systems deliver what they advocate for. Verily has been evaluated already on Pensive [107], DeepRM [106], and Aurora [71], and constitutes a first step towards alleviating current limitations.

11 SUMMARY

In this survey, we summarize research work regarding ML on computer systems and networking. Whether it concerns achievements or comes with limitations, we attempt to present an overall picture that exhibits current progress and exploits how ML can blend in various contexts and settings, and above all, motivate researchers to conceptualize and utilize ML in their field of research. We first formulate a taxonomy divided into distinct areas and sub-areas of expertise, in a quest to familiarize the reader with the greater picture. Eventually, we discuss each sub-domain separately. Per se, we formulate a short description of the problem and present the traditional approaches up to date and discuss the limitations of the traditional approaches. Then, we proceed to present the state-of-the-art of ML-based approaches and analyze significance of their results, limitations, and how far we are from an actual implementation in systems. We conclude the survey by discussing future directions to explore.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and the editor for their constructive comments and suggestions.

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 632–647.

[2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. 2006. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the Neural Information Processing Systems*. MIT, 1–8.

[3] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the ACM SIGMM Conference on Multimedia Systems*. ACM, 157–168.

[4] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay G. Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning video ABR algorithms to network conditions. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 44–58.

[5] Jamal N. Al-Karaki and Ahmed E. Kamal. 2004. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications* 11, 6 (2004), 6–28.

[6] Amir H. Ashouri, William Killian, John Cavazos, Gianluca Palermo, and Cristina Silvano. 2019. A survey on compiler autotuning using machine learning. *ACM Computing Surveys* 51, 5 (2019), 96:1–96:42.

[7] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. 2020. Generative models for effective ML on private, decentralized datasets. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

[8] Ron Avnur and Joseph M. Hellerstein. 2000. Eddies: Continuously adaptive query processing. In *Proceedings of the ACM International Conference on Management of Data*. ACM, 261–272.

[9] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. 1997. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking* 5, 6 (1997), 756–769.

[10] Andrzej Beben, Piotr Wisniewski, Jordi Mongay Batalla, and Piotr Krawiec. 2016. ABMA+: Lightweight and efficient algorithm for HTTP adaptive streaming. In *Proceedings of the International Conference on Multimedia Systems*. ACM, 2:1–2:11.

[11] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. 2019. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2019), 562–585.

[12] Daniel S. Berger. 2018. Towards lightweight and robust machine learning for CDN caching. In *Proceedings of the ACM Workshop on Hot Topics in Networks*. ACM, 134–140.

[13] Marcel Blöcher, Lin Wang, Patrick Eugster, and Max Schmidt. 2021. Switches for HIRE: Resource scheduling for data center in-network computing. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 268–285.

[14] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou. 2014. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 285–300.

[15] Justin A. Boyan and Michael L. Littman. 1993. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. Morgan Kaufmann, 671–678.

[16] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. 1994. TCP vegas: New techniques for congestion detection and avoidance. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 24–35.

[17] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based congestion control. *Communications of the ACM* 60, 2 (2017), 58–66.

[18] Surajit Chaudhuri. 1998. An overview of query optimization in relational systems. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Alberto O. Mendelzon and Jan Paredaens (Eds.), ACM, 34–43.

[19] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An efficient cost-driven index selection tool for microsoft SQL server. In *Proceedings of the International Conference on Very Large Data Bases*. Morgan Kaufmann, 146–155.

[20] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. Understanding TCP incast throughput collapse in datacenter networks. In *Proceedings of the ACM SIGCOMM Workshop on Research on Enterprise Networking*. ACM, 73–82.

[21] Federico Chiariotti, Stefano D'Aronco, Laura Toni, and Pascal Frossard. 2016. Online learning adaptation strategy for DASH clients. In *Proceedings of the International Conference on Multimedia Systems*. ACM, 8:1–8:12.

[22] Chia-Chen Chou, Aamer Jaleel, and Moinuddin K. Qureshi. 2017. BATMAN: Techniques for maximizing system bandwidth of memory systems with stacked-DRAM. In *Proceedings of the International Symposium on Memory Systems*. ACM, 268–280.

[23] Asaf Cidon, Assaf Eisenman, Mohammad Alizadeh, and Sachin Katti. 2016. Cliffhanger: Scaling performance cliffs in web memory caches. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 379–392.

[24] Maxim Claeys, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. 2014. Design and optimisation of a (FA)Q-learning-based HTTP adaptive streaming client. *Connection Science* 26, 1 (2014), 25–43.

[25] Maxim Claeys, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. 2013. Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming. In *Proceedings of the AAMAS Workshops*.

[26] Marc-Alexandre Côté and Hugo Larochelle. 2016. An infinite restricted boltzmann machine. *Neural Computation* 28, 7 (2016), 1265–1288.

[27] Carlo Curino, Subru Krishnan, Konstantinos Karanasos, Sriram Rao, Giovanni Matteo Fumarola, Botong Huang, Kishore Chaliparambil, Arun Suresh, Young Chen, Solom Heddaya, Roni Burd, Sarvesh Sakalanaga, Chris Douglas, Bill Ramsey, and Raghu Ramakrishnan. 2019. Hydra: A federated resource manager for data-center scale analytics. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 177–192.

[28] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. 2016. Job-aware scheduling in eagle: Divide and stick to your probes. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 497–509.

[29] Pamela Delgado, Florin Dinu, Anne-Marie Kermarrec, and Willy Zwaenepoel. 2015. Hawk: Hybrid datacenter scheduling. In *Proceedings of the USENIX Annual Technical Conference*. USENIX Association, 499–510.

[30] Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 77–88.

[31] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware cluster management. In *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 127–144.

[32] Arnaud Dethise, Marco Canini, and Srikanth Kandula. 2019. Cracking open the black box: What observations can tell us about reinforcement learning agents. In *Proceedings of the ACM SIGCOMM Workshop on Network Meets AI & ML*. ACM, 29–36.

[33] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, David B. Lomet, and Tim Kraska. 2020. ALEX: An updatable adaptive learned index. In *Proceedings of the International Conference on Management of Data*. ACM, 969–984.

[34] Mo Dong, Qingxi Li, Doron Zarchy, Philip Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting congestion control for consistent high performance. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 395–408.

[35] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC vivace: Online-learning congestion control. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 343–356.

[36] Thaleia Dimitra Doudali, Sergey Blagodurov, Abhinav Vishnu, Sudhanva Gurumurthi, and Ada Gavrilovska. 2019. Kleio: A hybrid memory page scheduler with machine intelligence. In *Proceedings of the International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 37–48.

[37] Thang Le Duc, Rafael A. García Leiva, Paolo Casari, and Per-Olov Östberg. 2019. Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM Computing Surveys* 52, 5 (2019), 94:1–94:39.

[38] Subramanya Dulloor, Amitabha Roy, Zheguang Zhao, Narayanan Sundaram, Nadathur Satish, Rajesh Sankaran, Jeff Jackson, and Karsten Schwan. 2016. Data tiering in heterogeneous memory systems. In *Proceedings of the European Conference on Computer Systems*. ACM, 15:1–15:16.

[39] Gil Einziger, Roy Friedman, and Ben Manes. 2017. TinyLFU: A highly efficient cache admission policy. *ACM Transactions on Storage* 13, 4 (2017), 35:1–35:31.

[40] Svante Ekelin, Martin Nilsson, Erik Hartikainen, Andreas Johnsson, Jan-Erik Mångs, Bob Melander, and Mats Björkman. 2006. Real-time measurement of end-to-end available bandwidth using kalman filtering. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*. IEEE, 73–84.

[41] Salma Emara, Baochun Li, and Yanjiao Chen. 2020. Eagle: Refining congestion control by learning from the experts. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 676–685.

[42] Jim Esch. 2015. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE* 103, 1 (2015), 10–13.

[43] Alaknantha Eswaradass, Xian-He Sun, and Ming Wu. 2005. A neural network based predictive mechanism for available bandwidth. In *Proceedings of the International Parallel and Distributed Processing Symposium*. IEEE Computer Society.

[44] Zubair Md. Fadlullah, Fengxiao Tang, Bomin Mao, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani. 2017. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys and Tutorials* 19, 4 (2017), 2432–2455.

[45] Vladyslav Fedchenko, Giovanni Neglia, and Bruno Ribeiro. 2018. Feedforward neural networks for caching: N enough or too much? *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (2018), 139–142.

[46] Alan Ford, Costin Raiciu, Mark Handley, Olivier Bonaventure, and Christoph Paasch. 2020. TCP extensions for multipath operation with multiple addresses. *RFC* 8684 (2020), 1–68. https://datatracker.ietf.org/doc/html/rfc6824.

[47] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, and Andrea Zanella. 2017. D-DASH: A deep q-learning framework for DASH video streaming. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (2017), 703–718.

[48] Jim Gettys and Kathleen M. Nichols. 2012. Bufferbloat: Dark buffers in the internet. *Communications of the ACM* 55, 1 (2012), 57–65.

[49] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association.

[50] Tomer Gilad, Neta Rozen Schiff, Philip Brighten Godfrey, Costin Raiciu, and Michael Schapira. 2020. MPCC: Online learning multipath transport. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies*. ACM, 121–135.

[51] Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert N. M. Watson, and Steven Hand. 2016. Firmament: Fast, centralized cluster scheduling at scale. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 99–115.

[52] Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*. JMLR.org, 1764–1772.

[53] Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. 1997. Index selection for OLAP. In *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society, 208–219.
[54] Pankaj Gupta and Nick McKeown. 2000. Classifying packets with hierarchical intelligent cuttings. *IEEE Micro* 20, 1 (2000), 34–41.
[55] Pankaj Gupta and Nick McKeown. 2001. Algorithms for packet classification. *IEEE Network* 15, 2 (2001), 24–32.
[56] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5 (2008), 64–74.
[57] Milad Hashemi, Kevin Swersky, Jamie A. Smith, Grant Ayers, Heiner Litz, Jichuan Chang, Christos Kozyrakis, and Parthasarathy Ranganathan. 2018. Learning memory access patterns. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1924–1933.
[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 770–778.
[59] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy H. Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association.
[60] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[61] Janey C. Hoe. 1996. Improving the start-up behavior of a congestion control scheme for TCP. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 270–280.
[62] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 187–198.
[63] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. 2020. Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys and Tutorials* 22, 3 (2020), 1686–1721.
[64] Ihab F. Ilyas, Volker Markl, Peter J. Haas, Paul Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the ACM International Conference on Management of Data*. ACM, 647–658.
[65] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew V. Goldberg. 2009. Quincy: Fair scheduling for distributed computing clusters. In *Proceedings of the ACM Symposium on Operating Systems Principles*. ACM, 261–276.
[66] Van Jacobson. 1988. Congestion avoidance and control. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 314–329.
[67] Akanksha Jain and Calvin Lin. 2013. Linearizing irregular memory accesses for improved correlated prefetching. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*. ACM, 247–259.
[68] Manish Jain and Constantinos Dovrolis. 2003. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions on Networking* 11, 4 (2003), 537–549.
[69] Manish Jain and Constantinos Dovrolis. 2004. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*. ACM, 272–277.
[70] Hasibul Jamil and Ning Weng. 2020. Multibit tries packet classification with deep reinforcement learning. In *Proceedings of the IEEE International Conference on High Performance Switching and Routing*. IEEE, 1–6.
[71] Nathan Jay, Noga H. Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3050–3059.
[72] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2014. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. *IEEE/ACM Transactions on Networking* 22, 1 (2014), 326–340.
[73] Sudarsun Kannan, Ada Gavrilovska, Vishal Gupta, and Karsten Schwan. 2017. HeteroOS: OS design for heterogeneous memory management in datacenter. In *Proceedings of the International Symposium on Computer Architecture*. ACM, 521–534.
[74] Konstantinos Karanasos, Sriram Rao, Carlo Curino, Chris Douglas, Kishore Chaliparambil, Giovanni Matteo Fumarola, Solom Heddaya, Raghu Ramakrishnan, and Sarvesh Sakalanaga. 2015. Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In *Proceedings of the USENIX Annual Technical Conference*. USENIX Association, 485–497.
[75] Phil Karn, Carsten Bormann, Gorry Fairhurst, Dan Grossman, Reiner Ludwig, Jamshid Mahdavi, Gabriel Montenegro, Joe Touch, and Lloyd Wood. 2004. Advice for internet subnetwork designers. *RFC* 3819 (2004), 1–60. https://datatracker.ietf.org/doc/html/rfc3819/.
[76] Yafim Kazak, Clark W. Barrett, Guy Katz, and Michael Schapira. 2019. Verifying deep-RL-driven systems. In *Proceedings of the ACM SIGCOMM Workshop on Network Meets AI & ML*. ACM, 83–89.

[77] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. 2013. MPTCP is not pareto-optimal: Performance issues and a possible solution. *IEEE/ACM Transactions on Networking* 21, 5 (2013), 1651–1665.
[78] Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. 2018. Neural networks for measurement-based bandwidth estimation. In *Proceedings of the IFIP TC6 Networking Conference*. IFIP, 460–468.
[79] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned cardinalities: Estimating correlated joins with deep learning. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*. www.cidrdb.org.
[80] Vadim Kirilin, Aditya Sundarrajan, Sergey Gorinsky, and Ramesh K. Sitaraman. 2020. RL-Cache: Learning-based cache admission for content delivery. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2372–2385.
[81] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*.Springer, 282–293.
[82] Kirill Kogan, Sergey I. Nikolenko, Ori Rottenstreich, William Culhane, and Patrick Eugster. 2014. SAX-PAC (scalable and eXpressive PAcket classification). In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 15–26.
[83] Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H. Chi, Ani Kristo, Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019. SageDB: A learned database system. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*. www.cidrdb.org.
[84] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the International Conference on Management of Data*. ACM, 489–504.
[85] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph M. Hellerstein, and Ion Stoica. 2018. Learning to optimize join queries with deep reinforcement learning. arXiv:1808.03196. Retrieved from https://arxiv.org/abs/1808.03196.
[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60, 6 (2017), 84–90.
[87] Miroslav Kubat. 2017. *An Introduction to Machine Learning, Second Edition*. Springer.
[88] Jim Kurose and Keith Ross. 2021. *Computer Networking: A Top-Down Approach* (8th ed.). Pearson.
[89] Karthik Lakshminarayanan, Anand Rangarajan, and Srinivasan Venkatachary. 2005. Algorithms for advanced packet classification with ternary CAMs. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 193–204.
[90] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215.
[91] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2018. Query optimization through the looking glass, and what we found running the Join Order Benchmark. *The VLDB Journal* 27, 5 (2018), 643–668.
[92] Qingxi Li, Mo Dong, and Philip Brighten Godfrey. 2015. Halfback: Running short flows quickly and safely. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 22:1–22:13.
[93] Rui Li, Bohan Zhao, Ruixin Chen, and Jin Zhao. 2020. Taming the wildcards: Towards dependency-free rule caching with FreeCache. In *Proceedings of the IEEE/ACM International Symposium on Quality of Service*. IEEE, 1–10.
[94] Wenjun Li, Xianfeng Li, Hui Li, and Gaogang Xie. 2018. CutSplit: A decision-tree combining cutting and splitting for scalable packet classification. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 2645–2653.
[95] Wenzhong Li, Han Zhang, Shaohua Gao, Chaojing Xue, Xiaoliang Wang, and Sanglu Lu. 2019. SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks. *IEEE Journal on Selected Areas in Communications* 37, 11 (2019), 2621–2633.
[96] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.
[97] Eric Liang, Hang Zhu, Xin Jin, and Ion Stoica. 2019. Neural packet classification. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 256–269.
[98] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md. Enamul Haque, Lingjia Tang, and Jason Mars. 2018. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 751–766.
[99] Wai-Xi Liu, Jun Cai, Qing Chun Chen, and Yu Wang. 2021. DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks. *The Journal of Network and Computer Applications* 177 (2021), 102865. https://www.sciencedirect.com/science/article/abs/pii/S1084804520303313.
[100] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.
[101] Jinhong Luo, Xijun Li, Mingxuan Yuan, Jianguo Yao, and Jia Zeng. 2021. Learning to optimize DAG scheduling in heterogeneous environment. arXiv:2103.06980. Retrieved from https://arxiv.org/abs/2103.06980.

[102] Bruce M. Maggs and Ramesh K. Sitaraman. 2015. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 52–66.
[103] Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. 2020. Themis: Fair and efficient GPU cluster scheduling. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 289–304.
[104] Zoubir Mammeri. 2019. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access* 7 (2019), 55916–55950. https://ieeexplore.ieee.org/document/8701570.
[105] Bomin Mao, Zubair Md. Fadlullah, Fengxiao Tang, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani. 2017. Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Transactions on Computers* 66, 11 (2017), 1946–1960.
[106] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the ACM Workshop on Hot Topics in Networks*. ACM, 50–56.
[107] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 197–210.
[108] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 270–288.
[109] Ryan Marcus and Olga Papaemmanouil. 2018. Deep reinforcement learning for join order enumeration. In *Proceedings of the International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. ACM, 3:1–3:4.
[110] Ryan C. Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A learned query optimizer. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1705–1718.
[111] Nicholas Mastronarde and Mihaela van der Schaar. 2011. Fast reinforcement learning for energy-efficient wireless communication. *IEEE Transactions on Signal Processing* 59, 12 (2011), 6262–6266.
[112] Nick McKeown, Thomas E. Anderson, Hari Balakrishnan, Guru M. Parulkar, Larry L. Peterson, Jennifer Rexford, Scott Shenker, and Jonathan S. Turner. 2008. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.
[113] H. Brendan McMahan and Daniel Ramage. 2017. Federated Learning: Collaborative Machine Learning without Centralized Training Data. (2017). Retrieved June 13, 2022 from https://ai.googleblog.com/2017/04/federated-learning-collaborative.
[114] Mitesh R. Meswani, Sergey Blagodurov, David Roberts, John Slice, Mike Ignatowski, and Gabriel H. Loh. 2015. Heterogeneous memory architectures: A HW/SW approach for mixing die-stacked and off-package memories. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. IEEE Computer Society, 126–136.
[115] Nithin Michael and Ao Tang. 2015. HALO: Hop-by-hop adaptive link-state optimal routing. *IEEE/ACM Transactions on Networking* 23, 6 (2015), 1862–1875.
[116] Pierre Michaud. 2016. Best-offset hardware prefetching. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. IEEE Computer Society, 469–480.
[117] Azalia Mirhoseini, Anna Goldie, Hieu Pham, Benoit Steiner, Quoc V. Le, and Jeff Dean. 2018. A hierarchical model for device placement. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.
[118] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device placement optimization with reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2430–2439.
[119] Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. 2014. Recursively cautious congestion control. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 373–385.
[120] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. JMLR.org, 1928–1937.
[121] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. JMLR.org, 1928–1937.
[122] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. 2020. Heterogeneity-aware cluster scheduling policies for deep learning workloads. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 481–498.
[123] Kyle J. Nesbit and James E. Smith. 2004. Data cache prefetching using a global history buffer. In *Proceedings of the International Conference on High-Performance Computer Architecture*. IEEE Computer Society, 96–105.

[124] Thuy T. T. Nguyen and Grenville J. Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials* 10, 1–4 (2008), 56–76.
[125] Jennifer Ortiz, Magdalena Balazinska, Johannes Gehrke, and S. Sathiya Keerthi. 2018. Learning state representations for query optimization with deep reinforcement learning. In *Proceedings of the 2nd Workshop on Data Management for End-To-End Machine Learning*. ACM, 4:1–4:4.
[126] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. 2013. Sparrow: Distributed, low latency scheduling. In *Proceedings of the ACM Symposium on Operating Systems Principles*. ACM, 69–84.
[127] Qiuyu Peng, Anwar Walid, Jaehyun Hwang, and Steven H. Low. 2016. Multipath TCP: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking* 24, 1 (2016), 596–609.
[128] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, Chen Meng, and Wei Lin. 2021. DL2: A deep learning-driven scheduler for deep learning clusters. *IEEE Transactions on Parallel and Distributed Systems* 32, 8 (2021), 1947–1960.
[129] Matthew Perron, Zeyuan Shang, Tim Kraska, and Michael Stonebraker. 2019. How I learned to stop worrying and love re-optimization. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 1758–1761.
[130] Meng Qin, Kai Lei, Bo Bai, and Gong Zhang. 2019. Towards a profiling view for unsupervised traffic classification by exploring the statistic features and link patterns. In *Proceedings of the ACM Workshop on Network Meets AI & ML*. ACM, 50–56.
[131] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the ACM International Conference on Machine Learning*. ACM, 873–880.
[132] Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, and Minlan Yu. 2015. Hopper: Decentralized speculation-aware cluster scheduling at scale. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 379–392.
[133] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. JMLR.org, 627–635.
[134] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2020. RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2260–2270.
[135] Constantin Sander, Jan Rüth, Oliver Hohlfeld, and Klaus Wehrle. 2019. DeePCCI: Deep learning-based passive congestion control identification. In *Proceedings of the ACM SIGCOMM Workshop on Network Meets AI & ML*. ACM, 37–43.
[136] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*. JMLR.org, 1889–1897.
[137] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv:1707.06347. Retrieved from https://arxiv.org/abs/1707.06347.
[138] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the ACM European Conference on Computer Systems*. ACM, 351–364.
[139] Du Shen, Xu Liu, and Felix Xiaozhu Lin. 2016. Characterizing emerging heterogeneous memory. In *Proceedings of the ACM SIGPLAN International Symposium on Memory Management*. ACM, 13–23.
[140] Manjunath Shevgoor, Sahil Koladiya, Rajeev Balasubramonian, Chris Wilkerson, Seth H. Pugsley, and Zeshan Chishti. 2015. Efficiently prefetching complex address patterns. In *Proceedings of the International Symposium on Microarchitecture*. ACM, 141–152.
[141] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.
[142] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. 2003. Packet classification using multidimensional cutting. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 213–224.
[143] Iraj Sodagar. 2011. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE MultiMedia* 18, 4 (2011), 62–67.
[144] Stephen Somogyi, Thomas F. Wenisch, Anastassia Ailamaki, Babak Falsafi, and Andreas Moshovos. 2006. Spatial memory streaming. In *Proceedings of the International Symposium on Computer Architecture*. IEEE Computer Society, 252–263.
[145] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.
[146] Ed Spitznagel, David E. Taylor, and Jonathan S. Turner. 2003. Packet classification using extended TCAMs. In *Proceedings of the IEEE International Conference on Network Protocols*. IEEE Computer Society, 120–131.

[147] Ashwin Sridharan, Roch Guérin, and Christophe Diot. 2005. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking* 13, 2 (2005), 234–247.

[148] Venkatachary Srinivasan, Subhash Suri, and George Varghese. 1999. Packet classification using tuple space search. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 135–146.

[149] Ajitesh Srivastava, Angelos Lazaris, Benjamin Brooks, Rajgopal Kannan, and Viktor K. Prasanna. 2019. Predicting memory accesses: The road to compact ML-driven prefetcher. In *Proceedings of the International Symposium on Memory Systems*. ACM, 461–470.

[150] Statista. 2019. Hours of Video Uploaded to YouTube Every Minute as of February 2020. (2019). Retrieved June 13, 2022 from https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/.

[151] Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. 2001. LEO - DB2's LEarning optimizer. In *Proceedings of the International Conference on Very Large Data Bases*. Morgan Kaufmann, 19–28.

[152] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the Conference of the Association for Computational Linguistics*. Association for Computational Linguistics, 3645–3650.

[153] José Suárez-Varela, Albert Mestres, Junlin Yu, Li Kuang, Haoyu Feng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2019. Feature engineering for deep reinforcement learning based routing. In *Proceedings of the IEEE International Conference on Communications*. IEEE, 1–6.

[154] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 272–285.

[155] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Neural Information Processing Systems*. 3104–3112.

[156] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1–9.

[157] Liansheng Tan, Cao Yuan, and Moshe Zukerman. 2005. FAST TCP: Fairness and queuing issues. *IEEE Communications Letters* 9, 8 (2005), 762–764.

[158] David E. Taylor. 2005. Survey and taxonomy of packet classification techniques. *ACM Computing Surveys* 37, 3 (2005), 238–275.

[159] David E. Taylor and Jonathan S. Turner. 2005. Scalable packet classification using distributed crossproducting of field labels. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 269–280.

[160] Guibin Tian and Yong Liu. 2012. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 109–120.

[161] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: The next generation. In *Proceedings of the European Conference on Computer Systems*. ACM, 30:1–30:14.

[162] Immanuel Trummer, Junxiong Wang, Deepak Maram, Samuel Moseley, Saehan Jo, and Joseph Antonakakis. 2019. SkinnerDB: Regret-bounded query evaluation via reinforcement learning. In *Proceedings of the International Conference on Management of Data*. ACM, 1153–1170.

[163] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A. Kozuch, Mor Harchol-Balter, and Gregory R. Ganger. 2016. TetriSched: Global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. In *Proceedings of the European Conference on Computer Systems*. ACM, 35:1–35:16.

[164] Kostas Tzoumas, Timos Sellis, and Christian Søndergaard Jensen. 2008. *A Reinforcement Learning Approach for Adaptive Query Processing*. Number 22. Institut for Datalogi, Aalborg Universitet, Denmark.

[165] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. 2017. Learning to route. In *Proceedings of the ACM Workshop on Hot Topics in Networks*. ACM, 185–191.

[166] Gary Valentin, Michael Zuliani, Daniel C. Zilio, Guy M. Lohman, and Alan Skelley. 2000. DB2 advisor: An optimizer smart enough to recommend its own indexes. In *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society, 101–110.

[167] Balajee Vamanan, Gwendolyn Voskuilen, and T. N. Vijaykumar. 2010. EffiCuts: Optimizing packet classification for memory and throughput. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 207–218.

[168] Jeroen van der Hooft, Stefano Petrangeli, Maxim Claeys, Jeroen Famaey, and Filip De Turck. 2015. A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 131–138.

[169] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems*.ACM, 18:1–18:17.

[170] Cisco VNI. 2018. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper. (2018). Retrieved 19 Feb 2019 from https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[171] Cong Wang, Amr Rizk, and Michael Zink. 2016. SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP. In *Proceedings of the International Conference on Multimedia Systems*. ACM, 1:1–1:12.

[172] Hao Wang and Baochun Li. 2017. Lube: Mitigating bottlenecks in wide area data analytics. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing*. USENIX Association.

[173] Hao Wang, Di Niu, and Baochun Li. 2018. Dynamic and decentralized global analytics via machine learning. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 14–25.

[174] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. 2018. Machine learning for networking: Workflow, advances and opportunities. *IEEE Network* 32, 2 (2018), 92–99.

[175] Ning Wang, Kin-Hon Ho, George Pavlou, and Michael P. Howarth. 2008. An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys and Tutorials* 10, 1–4 (2008), 36–56.

[176] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. 2019. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing*. USENIX Association.

[177] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. In *Proceedings of the Machine Learning*. 279–292.

[178] Keith Winstein and Hari Balakrishnan. 2011. End-to-end transmission control by modeling uncertainty about the network state. In *Proceedings of the ACM Workshop on Hot Topics in Networks*. ACM, 19.

[179] Keith Winstein and Hari Balakrishnan. 2013. TCP ex machina: Computer-generated congestion control. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 123–134.

[180] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 459–471.

[181] Kai Wu and Dong Li. 2021. Unimem: Runtime data management on non-volatile memory-based heterogeneous main memory for high performance computing. *Journal of Computer Science and Technology* 36, 1 (2021), 90–109.

[182] Kai Wu, Jie Ren, and Dong Li. 2018. Runtime data management on non-volatile memory-based heterogeneous memory for task-parallel programs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. IEEE/ACM, 31:1–31:13.

[183] Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. 2018. Gandiva: Introspective cluster scheduling for deep learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 595–610.

[184] Zhiyuan Xu, Jian Tang, Chengxiang Yin, Yanzhi Wang, and Guoliang Xue. 2019. Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1325–1336.

[185] Neeraja Jayant Yadwadkar, Ganesh Ananthanarayanan, and Randy H. Katz. 2014. Wrangler: Predictable and faster jobs using fewer resources. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 26:1–26:14.

[186] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Alexander Levis, and Keith Winstein. 2018. Pantheon: The training ground for Internet congestion-control research. In *Proceedings of the USENIX Annual Technical Conference*. USENIX Association, 731–743.

[187] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep unsupervised cardinality estimation. *Proceedings of the VLDB Endowment* 13, 3 (2019), 279–292.

[188] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 645–661.

[189] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 325–338.

[190] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys* 54, 6 (2021), 131:1–131:36.

[191] Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng, and Jin Zhao. 2019. Toward packet routing with fully-distributed multi-agent deep reinforcement learning. In *Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. IEEE, 1–8.

[192] Xiangyao Yu, Christopher J. Hughes, Nadathur Satish, and Srinivas Devadas. 2015. IMP: Indirect memory prefetcher. In *Proceedings of the International Symposium on Microarchitecture*. ACM, 178–190.

[193] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive congestion control for unpredictable cellular networks. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 509–522.

[194] Yuan Zeng and Xiaochen Guo. 2017. Long short term memory based hardware prefetcher: A case study. In *Proceedings of the International Symposium on Memory Systems*. ACM, 305–311.

[195] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in Mobile and wireless networking: A survey. *IEEE Communications Surveys and Tutorials* 21, 3 (2019), 2224–2287.

[196] Hong Zhang, Li Chen, Bairen Yi, Kai Chen, Mosharaf Chowdhury, and Yanhui Geng. 2016. CODA: Toward automatically identifying and scheduling coflows in the dark. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication*. ACM, 160–173.

[197] Ying Zheng, Ziyu Liu, Xinyu You, Yuedong Xu, and Junchen Jiang. 2018. Demystifying deep learning in networking. In *Proceedings of the Asia-Pacific Workshop on Networking*. ACM, 1–7.

[198] Chao Zhou, Chia-Wen Lin, and Zongming Guo. 2016. mDASH: A Markov decision-based rate adaptation approach for dynamic HTTP streaming. *IEEE Transactions on Multimedia* 18, 4 (2016), 738–751.

[199] Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning*. AAAI, 928–936.

[200] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. 2015. Can accurate predictions improve video streaming in cellular networks?. In *Proceedings of the International Workshop on Mobile Computing Systems and Applications*. ACM, 57–62.

Received 10 June 2021; revised 25 February 2022; accepted 28 February 2022