# Machine Learning in Real-Time Internet of Things (IoT) Systems: A Survey

Jiang Bian, *Member, IEEE*, Abdullah Al Arafat, *Graduate Student Member, IEEE*, Haoyi Xiong, *Senior Member, IEEE*, Jing Li, *Member, IEEE*, Li Li, *Member, IEEE*, Hongyang Chen, *Senior Member, IEEE*, Jun Wang, *Fellow, IEEE*, Dejing Dou, *Senior Member, IEEE*, and Zhishan Guo, *Senior Member, IEEE*

***Abstract*—Over the last decade, machine learning (ML) and deep learning (DL) algorithms have significantly evolved and been employed in diverse applications, such as computer vision, natural language processing, automated speech recognition, etc. Real-time safety-critical embedded and Internet of Things (IoT) systems, such as autonomous driving systems, UAVs, drones, security robots, etc., heavily rely on ML/DL-based technologies, accelerated with the improvement of hardware technologies. The cost of a deadline (required time constraint) missed by ML/DL algorithms would be catastrophic in these safety-critical systems. However, ML/DL algorithm-based applications have more concerns about accuracy than strict time requirements. Accordingly, researchers from the real-time systems (RTSs) community address the strict timing requirements of ML/DL technologies to include in RTSs. This article will rigorously explore the state-of-the-art results emphasizing the strengths and weaknesses in ML/DL-based scheduling techniques, accuracy versus execution time tradeoff policies of ML algorithms, and security and privacy of learning-based algorithms in real-time IoT systems.**

***Index Terms*—Deep learning (DL), Internet of Things (IoT), machine learning (ML), real-time systems (RTSs), scheduling.**

Manuscript received January 18, 2022; revised February 17, 2022; accepted March 14, 2022. Date of publication March 22, 2022; date of current version May 23, 2022. This work was supported in part by the National Science Foundation, USA, under Grant CNS–1948457, Grant CNS–1850851, Grant PPoSS-2028481, and Grant OIA–1937833. The work of Jiang Bian and Haoyi Xiong was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0110303. *(Jiang Bian and Abdullah Al Arafat contributed equally to this work.) (Corresponding authors: Zhishan Guo; Haoyi Xiong.)*

Jiang Bian was with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA. He is now with the Big Data Laboratory, Baidu Inc., Beijing 100193, China (e-mail: bianjiang03@baidu.com).

Abdullah Al Arafat, Jun Wang, and Zhishan Guo are with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA (e-mail: abdullah.arafat@knights.ucf.edu; jun.wang@ucf.edu; zsguo@ucf.edu).

Haoyi Xiong and Dejing Dou are with the Big Data Laboratory, Baidu Inc., Beijing 100193, China (e-mail: xionghaoyi@baidu.com; doudejing@baidu.com).

Jing Li is with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: jingli@njit.edu).

Li Li is with the Department of Computer and Information Science, University of Macau, Macau, China (e-mail: li.li@siat.ac.cn).

Hongyang Chen is with the Research Center for Intelligent Network, Zhejiang Laboratory, Hangzhou 311121, Zhejiang, China (e-mail: dr.h.chen@ieee.org).

Digital Object Identifier 10.1109/JIOT.2022.3161050

## I. Introduction

REAL-TIME systems (RTSs) design must have both functional and temporal correctness [1], [2]. Thus, RTSs are traditionally designed with temporally predictable and deterministic algorithms. For instance, before implementing an online scheduler, the regular real-time scheduling algorithms have to perform (exact or sufficient only) deterministic (finite time) offline feasibility (also known as schedulability) tests [1]. However, the feasibility test of the scheduling algorithms becomes highly complicated (in most cases intractable) with the underlying system heterogeneity and inter- and intra-dependent tasks [3]. Hence, until recently, RTS was restricted to only safety- and mission-critical systems, such as avionics, spacecraft, etc., with dedicated proprietary hardware platforms and simple task models.

Nonetheless, with the revolution of embedded cyber–physical systems and the Internet of Things (IoT) (thanks to the rapid advancement of hardware, software, and communication technologies)—RTS has been ubiquitously used in numerous domains, including healthcare, such as implantable devices, transportation, such as autonomous vehicles, smart-cities, such as smart grids, and industrial environments, such as drone, robots, etc. One fundamental similarity among these increasingly complex cyber–physical systems is that the systems are interacting with the physical world with excellent efficiency, leveraging a large number of onboard sensors. Thus, the systems require high computational resources to process the vast amount of diverse data from onboard sensors.

The emerging applications of RTS pose a couple of challenges: 1) it requires a heterogeneous hardware platform consisting of CPUs (multicore), GPUs, or specialized accelerators [4], which complicate the resource-sharing models of RTS and 2) the task dependency forms task chains with multiple arrival rates necessitates the complicated workload models, such as DAGs, GANG task models, etc. Scheduling such a workload using a deterministic feasibility test upon a heterogeneous hardware platform is tedious.

Consequently, the RTS community has become interested in data-driven approaches to deal with many diverse data sources in RTS applications over the last few years. Fortunately, machine learning (ML) and deep learning (DL) [e.g., deep neural network (DNN)] have extensively progressed and are employed in enormous applications unprecedentedly over the last decade. So, ML in RTS, especially the systems with lots of onboard sensors, has received significant attention from research communities. Unfortunately, merging these two prolific research domains poses several critical challenges for their unparalleled goals. In general, the ML research community prioritizes ML algorithms' efficiency or accuracy, while the temporal correctness of the algorithms has significantly less importance. Besides, the behavior of ML algorithms is not deterministic. In contrast, the algorithms' deterministic behaviors and temporal correctness are critical in RTS. So, before implementing ML algorithms in RTS, thorough research in ML algorithms concerning the RTS requirements are imperative.

Most emerging RTS devices, for example, autonomous vehicles, drones, etc., interact with the physical world through onboard sensors (e.g., cameras, radar, LiDAR, IMU, etc.). Thus, there is a high chance of a vicious attack on physical sensors or sensor data integrity/injection attack, eventually damaging the ML model. Therefore, ML algorithms should be malicious attack resilient and temporally deterministic to be employed in RTS. Another concern in end devices is that the intellectual properties, such as the architecture or parameters of the ML model, could be stolen through physically monitoring system behaviors, for example, I/O data throughput, memory-access patterns, EM signal spoofing, etc.

*Contributions:* In this literature survey paper, we rigorously review the current research works on the cross domain of RTS and ML that addresses the technological adaptation requirements mentioned above. We categorize the research works into the following directions.

1) *Adaptation of ML Algorithms in RTS:* We review the existing papers that address the challenges of adapting ML algorithms in RTS, categorizing via algorithmic techniques, such as model compression (e.g., pruning, quantization, knowledge distillation, etc.) and real-time pipeline of ML algorithms.
2) *ML Algorithms for Schedulability Analysis:* We review the papers that address the worst case execution time (WCET) estimation of real-time workloads, analyze the schedulability of a given workload for specific underlying hardware platforms guaranteeing the real-time constraints, and predict the system behavior (e.g., clairvoyance) through ML algorithms.
3) *Privacy and Security in Real-Time System:* We further review the works related to the privacy and security of RTS through deploying ML algorithms.
4) Finally, we present several open problems and potential applications of ML in the RTS that are yet to explore.

Throughout this article, we discuss the strengths and weaknesses of proposed/developed solutions and point out the research gap.

*Organization:* The remainder of this article is organized as follows. Section II discusses the background of RTSs, specifically the concepts related to the scheduling of RTSs and the learning algorithms commonly/potentially employed in RTSs. Section III comprehensively surveys the existing works on ML in real-time IoT. Section IV presents the applicability of ML algorithms in real-time IoT systems. Section V points out research gaps and proposes possible employment of ML in real-time IoT. Finally, this article concludes in Section VI.

## II. Background

In this section, we will briefly discuss the general properties of real-time IoT system workloads and commonly used ML/DL algorithms in real-time IoT systems.

### A. Real-Time IoT Systems

Real-time IoT systems are implemented with a collection of concurrent tasks in the underlying hardware resources. The RTS tasks (or workloads—we use these two terms interchangeably) need to be scheduled in the available system resources to meet the timing constraints (e.g., deadline, WCET, etc.) associated with each task. Hence, the system designers need to construct a workload model of the tasks characterizing the resources and timing requirements of the tasks.

Scheduling problems of a formally described workload model are traditionally tackled with scheduling algorithms that guarantee both functional and temporal correctness of the execution of the task set. The design of scheduling algorithms depends on the real-time workload model. The real-time workload models and corresponding scheduling techniques are well explored in the community [2], [5], [6]. Generally, there are three types of real-time workloads models based on the release patterns of a task instance—periodic, sporadic, and aperiodic tasks. According to Liu and Layland real-time task model [2], a periodic task upon a uniprocessor system is a task that releases its instances (jobs) after a specific time period. In addition, it is assumed that each task can release infinite instances during the runtime. Note that, depending on the relation between deadline and period of a task, the task can be classified as constraint-deadline (deadline less or equal to period) or implicit-deadline (deadline is equal to period) task.

In the sporadic task model, the jobs of the task can release at any time, maintaining a minimum job separation time (minimum separation time is also referred to as *period* for sporadic model). A job of an aperiodic task can arrive at any time, and there is no periodicity/minimum separation of jobs. Also, the aperiodic task can be hard-aperiodic tasks where the released job has a deadline. Furthermore, depending on the inter- and intra-dependency of the jobs, more sophisticated workload models are developed, such as graph-based workload models (DAGs, Digraph, etc.,—a comprehensive survey on graph workload models [6]), Gang models [7], etc.

The design of real-time scheduling algorithms has two key steps: 1) offline verification/certification and 2) online scheduling strategy. Most scheduling problems are known as *NP-Hard* problems in a strong sense due to the complexity (intractability) of the offline certification (schedulability test) of scheduling algorithms. Therefore, the algorithms are typically designed with approximation and relaxation, maintaining a *sufficient only* condition with a lesser (e.g., pseudopolynomial) time complexity of the schedulability test. The scheduling algorithms are designed for two different types of priority: 1) static priority (task-level fixed priority), where the priority of each task in the task set is fixed and 2) dynamic priority (job-level fixed priority), where the priority of a task changes in each job instance depending on the released jobs of other tasks available in the queue. Real-time scheduling algorithms include preemptive (active task instance can be interrupted by a newly released higher priority task instance) or nonpreemptive algorithms, such as earliest deadline first (EDF) [5], rate monotonic (RM) [2], deadline monotonic (DM) [8], etc.

The workload scheduling complexity increases when upgraded from uniprocessor to multiprocessor platform. However, the uniprocessor scheduling algorithms can still use in a multiprocessor system with substantially modification techniques, such as global scheduling, partitioned scheduling, etc. [1]. Besides scheduling complexity, assessing the precise WCET of the jobs is also complicated. In most cases, it is very challenging to evaluate the exact WCET due to either unavailability of the system architecture for intellectual property reasons or the analysis complexity of deterministic WCET. Therefore, the estimated WCET is very pessimistic (often add a safety margin to the estimated WCETs), resulting in poor system utilization.

In modern autonomous systems, the systems typically have different criticality levels based on the systems' safety requirements. For instance, connected and autonomous vehicles (CAVs) have several system criticality requirements following the safety standards, such as ISO26262—the safety-criticality levels, such as anti-lock braking system, steering, engine controller, should have higher priority than the infotainment system, A/C, etc. To address the different criticality levels of these autonomous systems, Vestal [9] proposed a mixed-criticality system (MCS) model, which also improves the system utilization assigning different WCETs to a task for different system criticality levels (a comprehensive review on MCS is presented in [10]). All tasks are executed in a low-critical mode (regular operating mode) with the smallest WCET values in regular operation. Suppose the system fails to meet the deadline or overexecutes a high-critical task, then the system switches to the higher system critical level by graceful degradation or dropping of low-criticality tasks. In addition, (typically) only high-critical tasks have a timing guarantee in the higher criticality system mode. The MCS task set uses relatively low and optimistic WCET in regular operating mode. Hence, it is possible to derive a data-driven WCET for the task set, specifically for regular system operation mode.

### B. Machine Learning

ML includes a wide range of algorithms from end-to-end problem-solving algorithms to specific feature extraction algorithms [11]–[15]. ML algorithms are typically categorized into three directions: 1) supervised learning [16], [17]; 2) unsupervised learning [18], [19]; and 3) reinforcement learning [20], [21], which depend on the interaction or feedback between the learning algorithms and the learning systems. There are also a huge amount of ML algorithms beyond these three mainstream scopes (or interdisciplinary ones), which are widely used/adopted in real-time learning systems for some specific tasks. For example, meta learning for hyperparameter tuning on real-time embedded systems [22], real-time traffic classification through semisupervised learning [23], and real-time inference and training for DL [24]. Apart from the classical categorization of ML algorithms, we purposely divide the ML algorithms applied in real-time IoT systems into two branches, which are statistical learning algorithms and neural network-based learning/DL algorithms. The motivation behind is to comprehensively review the previous efforts of applying learning algorithms in RTSs by task complexity and data complexity. Since the data streams in modern real-time IoT systems are increasingly large-scale, dynamic, and heterogeneous, some of the traditional statistical learning strategies/tools are unable or inefficient to handle the complicated scenarios, which frequently occur in nowadays autonomous driving systems, security robots, online signal processing, etc. For example, the family of support vector machines [25] (SVMs) (e.g., kernel SVMs) are barely used in real-time object detection/recognition in autonomous driving systems due to its shallow architecture and inflexibility compared to the (deep) neural network [26], where any marginal increase of accuracy matters in such case for safety issue and (deep) neural network benefits from its ability to tackle large-scale data, and complicated tasks can beat SVMs in terms of accuracy under most circumstances [27]–[29]. However, there is no free lunch for ML. For some cases in real-time IoT systems (e.g., real-time task scheduling), statistical learning algorithms (comparably shallow one) are often adopted due to their characteristics of off-the-shelf and easy-to-train [30], [31] which save a lot of time when handling the "lite" and urgent tasks.

It is important to note that other than commonly/strictly defined ML algorithms, we discovered a rich amount of not clear-cut "learning" algorithms which are increasingly adopted in modern real-time IoT systems. These algorithms most lie in evolutionary computing [32], [33], stochastic search [34], [35], and other optimization algorithms (e.g., expectation–maximization algorithms [36] and augmented Lagrangian method [37]) which are rarely recognized as traditional ML algorithms.

As shown in Fig. 1, we also summarize the popular branches of ML algorithm in real-time IoT systems with a taxonomy. Note that the taxonomy does not cover all the branches of ML and may differ from the structure of other taxonomies, especially in ML domains, since we mainly adopt the branches fulfilling the real-time constraints. In the following sections, we will briefly review the most representative and commonly used learning algorithms in real-time IoT systems in the above-mentioned three aspects.

*1) Statistical Learning Algorithms:* As we investigated, the statistical ML algorithms have been widely used in real-time IoT systems for solving some classic problems, such as classification [38], [39], regression [40], [41], and clustering [42], [43]. By incorporating prior knowledge and entropy metric, correlation analysis, inherent statistical structures of input data, and nonlinear relations, statistical ML algorithms are easy to deploy [44], interpretable [45], [46], and trustworthy [47] in some of the real-time IoT applications ranging from traditional real-time scheduling systems [48], [49] to

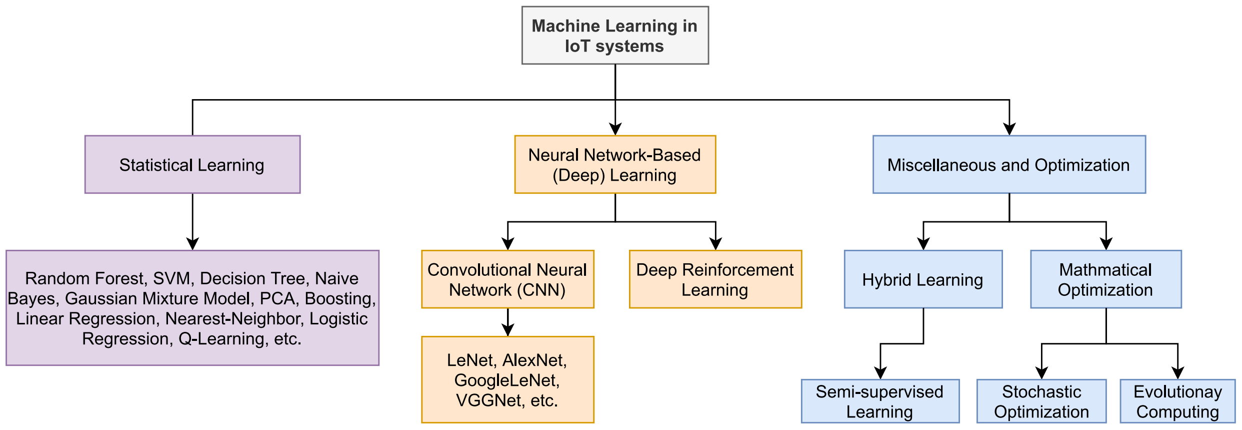

Machine Learning in IoT systems
Statistical Learning
Neural Network-Based (Deep) Learning
Miscellaneous and Optimization
Random Forest, SVM, Decision Tree, Naive Bayes, Gaussian Mixture Model, PCA, Boosting, Linear Regression, Nearest-Neighbor, Logistic Regression, Q-Learning, etc.
Convolutional Neural Network (CNN)
Deep Reinforcement Learning
LeNet, AlexNet, GoogleLeNet, VGGNet, etc.
Hybrid Learning
Mathmatical Optimization
Semi-supervised Learning
Stochastic Optimization
Evolutionay Computing

Fig. 1. Some popular ML branches in real-time IoT systems.

modern real-time IoT systems [50]–[52]. The family of these algorithms includes decision trees [53], random forest [54], Gaussian mixture model [55], naive Bayes [56], linear regression [57], logistic regression [58], SVM [25], boosting [59], nearest-neighbor methods [60], $Q$-learning [61], principal component analysis (PCA) [62], and so on. However, the main drawback of these statistical ML algorithms is straightforward, where the overall performance significantly degrades when either the complexity of tasks or the scale of data dramatically increases [26], [63]. We will systematically review and summarize the employment of statistical ML algorithms in real-time IoT systems in Section III.

*2) Neural Network-Based (Deep) Learning Algorithms:* On top of a wide variety of commonly used ML algorithms, neural network architectures play an important role in modern learning-based RTSs and applications. One most employed class of ML algorithms exploits neural network architecture and stack several layers of the neural network well known as DL, or DNN [64]–[66]. Frequently used neural network architectures include fully connected layers (FC-layers) [67], convolutional neural networks (CNNs) [68], recurrent neural networks (RNNs) [69], residual networks [27], etc. Among them, CNN and its variation with other networks are the most popular and highly used DL strategy in IoT systems, such as computer vision [70]–[72], natural language processing (NLP) [73], [74], activity recognition [75], [76], etc. We will briefly introduce two of the most popular neural network-based ML applications and their variations in the following sections.

*a) Convolutional neural networks:* CNN is a class of DL algorithms, highly used in high-dimensional data sets, such as images to extract low-dimensional latent space representations and location-invariant features [77]. CNN is the building block of the famous DL architectures, such as LeNet [78], AlexNet [79], GoogLeNet [80], VGGNet [81], etc. CNN consists of several types of layers, such as convolution layers, pooling layers, activation layers, fully connected layers, etc. As one of the most commonly used modules in DL/DNN, the real-time characteristics and resource consumption are the prime concerns when applying it to modern real-time IoT (embedded) systems. We will discuss the current disadvantages and the corresponding solutions for deploying CNN on real-time IoT systems in Section III-B1.

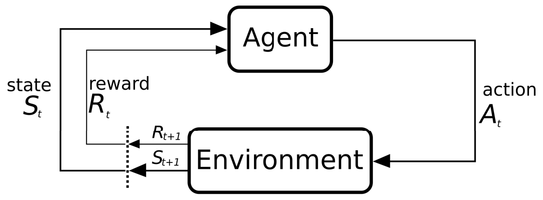


Fig. 2. Reinforcement learning framework.

*b) Deep reinforcement learning:* Reinforcement learning is a branch of ML technique, which is designed to solve problems via a feedback system, including rewards and penalties. The so-called agent in reinforcement learning moves through several states in an environment to achieve a predefined final state, as illustrated in Fig. 2. In the moving process, the agent exploits past experience and explores new states to achieve its goal. Through trial and error (penalties versus rewards), the agent will form the final solution of the problem. The solution consists of a series of the optimal sequence of states in which the accumulated sum of rewards is maximized.

However, reinforcement learning intends to embrace the hug of DL in nowadays real-time IoT applications due to its limitation on large-scale dynamic data environment [21], [82]. As we aforementioned, (deep) neural networks can be used to approximate specific function, which is especially useful in RL when the space of states or actions are too broad to be fully acknowledged. In specific, a neural network also be capable of approximating a value function, or a policy function. In other words, neural networks are able to learn mapping states to values. Instead of storing, indexing, and updating the mapping information in a lookup table, which is difficult for the large-scale problem, we train a neural network with samples in the space of state or action to learn the best strategy to achieve the goal of the learning process. Moreover, in deep reinforcement learning, convolutional networks are usually used to recognize an agent's state when the input is visual; e.g., wildlife tracking using deep convolutional UAV [83] in Fig. 3. That is, the UAV leverages the target image caption as the reward for movement guiding and wildlife tracking.

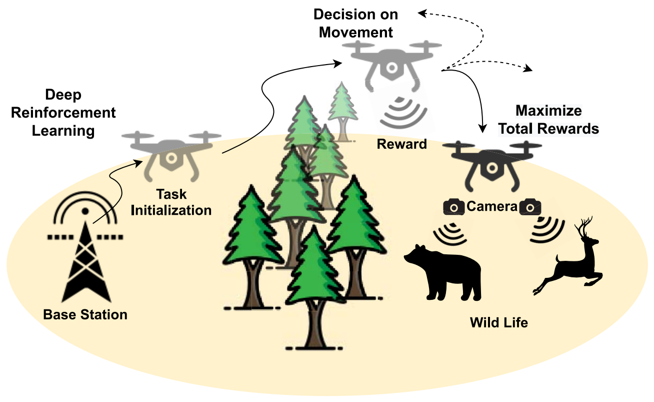


Fig. 3. Deep convolutional agent on UAV.

Rather than the above tasks in ML, deep reinforcement learning is also a powerful tool to solve combinatorial optimization and scheduling problems. These problems are often challenging in RTSs and IoT applications, e.g., learning near-optimal schedules when the RTS is not known in advance [84], resource protection, and real-time detection [85]. Another branch of representative applications which massively involve deep reinforcement learning is wireless communication, especially 6G [86], [87]. Since the wireless communication environment in modern IoT systems is highly dynamic and complex, the traditional ML algorithms confront the problem of heavy and inefficient mathematical computations, while deep reinforcement learning is capable of sustaining reliable wireless connectivity for the networks by learning the environment dynamics. We will further review the employment of deep reinforcement learning in Section III-A.

*3) Other Miscellaneous and Optimization Algorithms:* Beyond the scope of typical ML algorithms, we can also observe the rising of hybrid learning strategies and optimization algorithms in modern real-time IoT systems. For hybrid learning strategies, semisupervised learning [88], [89] is one of the popular directions. In semisupervised learning, we intend to form a supervised learning algorithm leveraging labeled data augmented by unlabeled data. The amount of unlabeled or partially labeled data is usually bigger than the amount of labeled data, since the latter is more expensive and difficult to obtain. Thus, the goal is to overcome one of the problems of supervised learning (also the unsupervised learning, where its application spectrum is limited)—having not enough labeled data. By adding cheap and abundant unlabeled data, we are hoping to build a better model than using supervised learning alone. Although semisupervised learning sounds like a reasonable approach, the practical employment is constrained by certain assumptions (manifold, cluster, or smoothness assumption [90]), which are more likely to be violated in RTS settings. For example, when handling the real-time streaming data, semisupervised learning would confront the issue of false self-training [23], [91] (mistake can re-enforce themselves) due to the fact that we rarely observe the true label especially in a real-time manner so as to be trapped into a wrong direction of learning (farther and farther from the true manifold of data itself).

Another branch is the well-known optimization strategies (also known as mathematical optimization) [92]–[94] which are not typically categorized into traditional ML algorithms. Although we often rely on them to solve the ML problems, e.g., stochastic optimization [95], [96] for backpropagation in training neural networks, the optimization algorithms could be independent of ML and provide a solid guide or approximation for the objectives in real-time IoT systems. As one of the representatives, evolutionary computing [32] has some of the practical advantages to be employed especially in real-time scheduling, which include the flexibility of the optimizing procedure, as well as their ability to self-adapt the search for optimum solutions on the fly. Specifically, each new generation is produced by stochastically removing less desired solutions and introducing small random changes, where this mechanism is naturally suitable for some extensive and creative searching, e.g., generate schedulability test [97], [98] or response time analysis [99]–[101] in real-time IoT systems. With such an evolutionary optimization algorithm, we can automatically explore the possible formation of schedulability tests which saves a lot of effort by manual checking and proofing. We will further review the related work which involves these miscellaneous and optimization algorithm algorithms in the following sections.

## III. ML in Real-Time IoT Systems

In this section, we will review the existing and potential of ML, DL, and miscellaneous algorithms to employ in real-time IoT systems. These algorithms range from almost all the popular branches in Section II-B and we are not going to dive into any specific branch but with a well-designed taxonomy of employment to showcase the relationship between ML algorithms and real-time IoT systems. Note that although hardware design also involves many ML-based algorithms and plays an important role in a real deployment, we mainly focus on the software level due to our core expertise and limited space. To fully investigate the entire paradigm, we initialize the discussion from two key components in the real-time IoT systems, which are *ML-based learning algorithm* and *real-time scheduler*. For the learning algorithm, it could play the role of either assisting the operation of the IoT system in a real-time manner or a target task to be scheduled to achieve the system level real-time and other optimization objectives. Simultaneously, the real-time scheduler is in charge of scheduling the task to guarantee the characteristics of hard/soft real time in IoT systems, in which the learning algorithms could enhance the efficiency and efficacy of the scheduling process. For specific applications, the real-time scheduler on the contrary can guide the learning algorithm to generate valid and affordable (by computation resource in system) solutions with a strong real-time guarantee. The interaction between these two components derives three mainstream of integration summarized as 1) ML-based learning algorithms for real-time scheduling; 2) adaptation of ML-based learning algorithm to make it schedulable; and 3) rising security issues when involving ML-based learning algorithms in real-time IoT systems. As illustrated in Fig. 4, a horizontal tree-like structure shows the

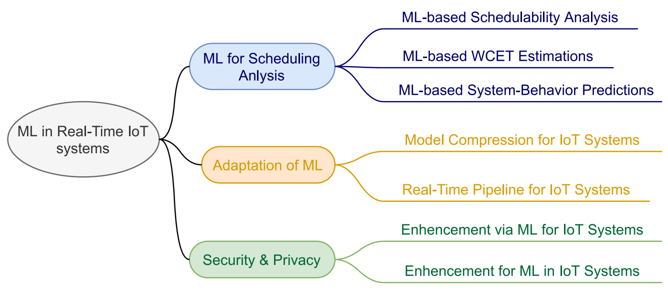

ML in Real-Time IoT systems
ML for Scheduling Anlysis
ML-based Schedulability Analysis
ML-based WCET Estimations
ML-based System-Behavior Predictions
Adaptation of ML
Model Compression for IoT Systems
Real-Time Pipeline for IoT Systems
Security & Privacy
Enhencement via ML for IoT Systems
Enhencement for ML in IoT Systems

Fig. 4. Tree structure of ML employment.

hierarchical categorization of ML employment in real-time IoT systems. Note that security issues increasingly draw attention in recent studies in terms of a variety of information leakage in real-time IoT systems by learning-based strategies or side-channel attacks [102], [103]. Thus, we separately review the security issues in real-time IoT systems on the top of the predefined categories. For the remaining parts of this section, we will discuss each branch with its leaves from top to bottom in descending order of the amount of related literature. The branch section will explore the employment and applications, which consists of the most popular and exploited leaf topics as follows.

1) ML for scheduling analysis in real-time IoT systems.
   a) ML-based schedulability analysis.
   b) ML-based WCET estimations.
   c) ML-based system-behavior prediction.
2) Adaptation of ML in real-time IoT systems.
   a) Model compression for real-time performance.
   b) Real-time pipeline.
3) Security of real-time IoT systems.

For each topic, we will discuss the motivation, scope, technical details, contribution, and remaining challenges of the related ML algorithms and the design of real-time IoT systems.

## A. ML for Scheduling Analysis in Real-Time IoT Systems

Real-time scheduling problems on modern hardware (e.g., heterogeneous and multiprocessor platform) are highly intractable and often *NP-Hard* in a strong sense. Traditional scheduling algorithms (e.g., dynamic and fixed-priority algorithms) are (mostly) approximate, yet often with high time complexity [104]. To tackle the scheduling problems, ML gets attention from the research community as early as the pioneering work of Hopfield and Tank [105] which uses neural networks for optimization purposes. The main challenge of using neural networks in real-time scheduling problems lies in the mapping of real-time scheduling constraints into the neural network setup. Few early attempts to map the real-time scheduling problem into neural networks to solve the scheduling problems are [106]–[113]. In the case of mapping the scheduling problem to the ML framework, we need to answer several questions.

1) How to design a data set (e.g., input–output pairs) using scheduling constraints for the ML framework?
2) How to choose a suitable ML framework or DNN architecture for the specified problem?
3) How to assign priority to the tasks for efficient model training and inference?

Lee *et al.* [114] proposed ML-based scheduling of fixed-priority task model. Guo and Baruah [115] developed a single-layer RNN model for a real-time scheduling problem on a uniprocessor system. Besides considering scheduling problems, ML-based schedulability analysis, WCET estimation, and RTS behavior prediction are getting attention from the research community. Although these three problems are interrelated, such as schedulability analysis requires WCET of each task and RTS behaviors are highly dependent on the system scheduling policies, analyzing each problem is tedious for large systems. We will discuss the related works in these three directions in the following sections.

*1) ML-Based Schedulability Analysis:* In RTSs, schedulability analysis of a scheduling algorithm has to perform before the system's runtime to guarantee the algorithm's timing correctness for a task set. The schedulability analysis is usually performed based on each task's WCET in the system. A general requirement of schedulability analysis of scheduling algorithms is determinism. So, the complexity or hardness of schedulability analysis is a significant concern in designing the scheduling algorithms. Moreover, the complexity of schedulability analysis increases with the increment of the cores/processors in the system.

In fact, due to the SWaP (size, weight, and power) limitations and high computational resource requirements of modern real-time IoT, the real-time chip design accelerates the urge to move from single processor to multiprocessors system (e.g., multiprocessor-system-on-a-chip (MPSoC) [4]). The multiprocessor system's schedulability analysis is highly complicated and, in most cases, *NP-Hard* or *NP-Hard in a strong sense* problem. It also becomes erratic if that schedulability analysis is derived from conventional ways such as response time analysis [116]-based schedulability analysis. To this end, researchers have become interested in mechanized schedulability analysis [117] rather than exact analysis. Dziurzanski *et al.* [118] used evolutionary algorithms to semi-automate *response time analysis* technique for schedulability analysis. There are some initial results on mechanized schedulability analysis; however, ML-based schedulability is still relatively unexplored. ML-based schedulability analysis would not give the exact schedulability of the scheduling algorithms. It is also essential to analyze the feasibility and reliability of ML-based schedulability analysis.

In contrast to RTSs' exact parameterized scheduling problems, real-time routing scheduling problems are usually formulated as the distribution of latency of each pair of nodes in the network. A possible shortest path for a source node to a destination node can be easily found using the Dijkstra algorithm [119] for worst case latency along the path of each intermediate edge. However, such an approach is very pessimistic. Recently, Agrawal *et al.* [120] proposed an RL-based routing algorithm constructing a *Q*-table using an optimal routing table of the network, and then they dynamically update the *Q*-table if any changes (e.g., add/drop of intermediate nodes) occur in the routing table during runtime. The real-time scheduling problem, in general, becomes a large search problem as system workload increases. RL performs efficiently for large search space problems, and Bo *et al.* [121] formulated the online scheduling problem for a system with aperiodic workloads using deep-RL. They intuitively modeled each job as an agent in the RL framework and formulated the scheduling decision problem as a Markov game.

*2) ML-Based WCET Estimations:* The precise WCET calculation requires the process's executable file (e.g., binary code, source code, intermediate code, etc.) and detailed knowledge of the target system's microarchitecture (e.g., cache, pipeline, branch predictor, etc.) for static analysis. In static analysis techniques, the (safe) WCET is estimated without executing the program leveraging the detailed system architectural knowledge. Therefore, the precise WCET estimation of the processes or tasks has become extremely difficult either the hardware architecture (e.g., MPSoCs [4]) becomes too complex to design a static analysis model or the unavailability of architectural details for intellectual property reasons. Besides static analysis techniques, two other traditional analytical methodologies, such as end-to-end measurement, and hybrid analysis techniques [122] (the interested reader may refer to [123] for a detailed survey on existing WCET estimation tools) are used to determine the WCETs.

In contrast to static analysis techniques, the end-to-end measurement techniques execute the process for several sets of input (without knowing system architecture) and collect the execution time. Then, WCET is chosen as the maximum observable execution time or uses statistical extrapolation with the addition of a safety margin to mitigate the lack of confidence in the measurement process. One critical drawback of measurement-based estimation is the code coverage problem—it is highly difficult to find inputs that cover all basic blocks of the target process. In hybrid analysis methods, the WCET of basic blocks of processes is usually estimated using measurement-based techniques. The WCET of the whole process is estimated using a static analysis tool (e.g., IPET [123]). However, these static analysis tools are very pessimistic, and the drawback of measurement-based estimation exists. Therefore, a more dynamic approach for estimating the WCET is necessary. An alternating approach of WCET estimation using ML framework with few early results are already proposed by [124]–[127]. Huybrechts *et al.* [124] developed regression algorithms and DL algorithms [125]-based WCET estimation methods for a hybrid scenario. In ML-based WCET estimation, an ML-based timing model is developed in the learning phase, and then the model is used to determine the timing of basic blocks. In the second phase, modified static analysis tools are used to find the timing of the whole control flow graph of the target process. Although ML-based approaches remove the drawback of measurement-based approaches, the ML model does not guarantee the perfect timing model of the system architecture. Therefore, these methods are not applicable in safety-critical hard-RTSs.

Vestal [9] presented a varying WCET-based scheduling technique called MCSs to avoid very pessimistic WCETs of tasks in complex systems. As the WCET estimation became difficult, the MCSs used different levels of WCETs based on the criticality levels to improve the average-case performance of the system. In these mixed-critical or multiple mode systems, the system designer has the freedom to choose an optimal WCET value for the lower critical task. So, the probabilistic WCET estimation became popular for low-critical tasks. A comprehensive survey of probabilistic worst case timing analysis is given in [128]. In fact, the system complexity affects other system parameters, such as the period of the tasks. So, it is often important to measure the run-time period of the tasks for better dynamic WCET estimation. Vădineanu and Nasri [40] developed regression algorithms-based run time period estimation methods for real-time tasks in complex systems.

*3) ML-Based System Behavior Prediction:* Real-time IoT applications are often used in safety-critical systems—a task missing a deadline can be catastrophic for the system and endanger human lives. Hence, hard RTSs are designed with deterministic behavior to guarantee the task meets every deadline. The safety-critical systems are traditionally designed as MCSs [9] or multimodel systems [129]. In MCSs, the system may switch its mode to different safety levels depending on the system behavior in runtime. Typically, the system is unaware of such a mode switch event prior to the occurrence (nonclairvoyant). Therefore, the scheduling algorithms for these scenarios incur significant overhead (e.g., dramatic increases of system workload demand or execution due to larger WCET's in higher critical levels) for the consideration of sudden mode-switch instances. It is obvious that clairvoyant or semi-clairvoyant scheduling algorithms perform better than the nonclairvoyant algorithms [130]. In clairvoyant scheduling algorithms, the scheduler assumes that the system mode-switch instant is known before the runtime. In contrast, in semi-clairvoyant algorithms, the mode-switch instant is known at the release instant of the job that initiates the system mode-switch to the higher criticality level. However, there is a practical implementation of the clairvoyance and semi-clairvoyance system yet. Recent work on quarter-clairvoyance (mode-switch instant is predicted in between the release instant of mode-switch initiator job and the mode-switch instant of the system), Pythia-MCS [131], leverages the I/O data throughput of the system. In Pythia-MCS, a statistical mode-switch instant detector is developed based on data traffic through I/O buses and both experimentally on the practical platform and analytically shows that Pythia-MCS performs better than the nonclairvoyant systems. However, Pythia-MCS did not use any ML in their design, it may be possible to use the ML framework to improve the predictability further than the quarter-clairvoyance.

## *B. Adaptation of ML in Real-Time IoT Systems*

Modern real-time IoT systems increasingly adopt the family of DNNs or DL to embrace the explosive growth of data scale and problem complexity in the big data era. Applying DNNs can drastically raise the performance of a wide range of applications than using statistical learning algorithms, e.g., accuracy in image recognition and feasibility of decision in autonomous control. However, one of the most significant challenges is that the real-timeness of the DNNs deployment is hard to be controlled and guaranteed. Since most of the DNNs are designed and developed on large-scale computing platform with powerful GPU clusters for specific performance boosting, the traditional DNNs is not available for strict timing requirement when applying to resource-constrained embedded RTSs, which is nowadays' trending environment in mobile and autonomous IoT systems. To handle the adaptability issue of DNNs, two representative directions for overhead mitigation are come up with in previous studies which are 1) compressing the model for implementation speed-up and 2) optimizing the system-wise pipeline for real-timeness. We will summarize the related works and discuss the contributions from these two directions in the following sections.

*1) Model Compression for Real-Time IoT Systems:* DNNs with DL bring a revolution in the broad domain of computer vision and NLP. As one of the most powerful tools, DNNs have a huge impact on the standard process of industry practices, where the classical two-stage process (i.e., training and inference) is widely adopted.

As aforementioned, we design a specific DNN model for the problem and train the model accordingly with the data set available, where the training process may take a long time (e.g., tens of hours or even a few weeks) on a GPU or a cluster of high-performance CPU. After the training process, we deploy the model in the target working environment where the stream of data is fed into the model for real-time inference. The output we obtained either is used as the final result or as the intermediate result for the downstream systems. However, the applications, e.g., autonomous car, and search engines nowadays require much less latency than before, which means the DL inference is required to be lightning-fast, usually less than tens of milliseconds for each output. Thus, different from the traditional academical focus on model training, the real-time IoT system takes more consideration on the inference speed, which brings an acceleration on DNN inference from the hardware and software aspects. In this work, we mainly investigate the software solution in real-time IoT systems.

From the algorithm perspective, model compression is one promising and commonly used method to decrease the latency of DNN inference and DRAM footprint. It is easy to fit compressed models in on-chip SRAM cache rather than off-chip DRAM memory and these models can help the DNNs work on mobile devices and other stream-data-based applications, especially the inference speed, memory size, and the communication bandwidth are constrained hardly. Fully connected layers are known to be overparameterized in most state-of-the-art DNN architectures. A lot of previous research has focused on compressing FC layers, either by bucketing connection weights (pseudo) randomly using a hash function or by vector quantization. Network-in-Network is proposed to replace FC layers with global average pooling, with an additional linear layer added at the top for better transferability. Benchmarked on CPU, desktop GPU, and mobile GPU, deep compression yields 30× to 50× more compact AlexNet and VGG-16 models that have 3× to 4× layerwise speedup and 3× to 7× higher energy efficiency, all without loss of accuracy on ImageNet. There are several techniques to reduce network size, for example, pruning the inference networks [132], quantization of network parameters to avoid floating-point operations [133], dropout of less important neurons [134], etc. We will summarize the main branch of these model compression techniques and discuss the advantages and the disadvantages of applying them in real-time IoT systems in Table I.

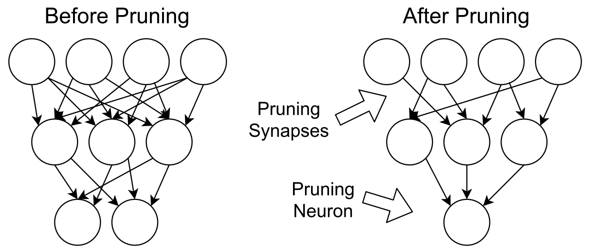


Fig. 5. Pruning on synapses and neuron.

1) *Pruning:* Pruning [132], [138]–[146] intends to remove redundant, unnecessary connections that are not sensitive to performance to compress the model (decrease the number of parameters). This not only helps reduce the overall model size but also saves on computation time and energy. As shown in Fig. 5, the number of synapses and neurons are in some degree reduced in the pruning process.
2) *Quantization:* The weight parameters are usually stored as 32-bit floating-point numbers in DNNs. Quantization is one way to represent these weight parameters through reducing the number of bits [147]–[151]. The weight parameters can be customized to 16-bit, 8-bit, 4-bit, or even with 1-bit. Since the number of bits is decreased, the size of the DNN can be significantly shrunken.
3) *Knowledge Distillation:* Knowledge distillation [152]–[155] is often used in model training with large-scale data set. It is natural thinking to transfer the originally large and complicated model (well trained) to a smaller and compact one. The originally large model is the so-called teacher network, while the transferred smaller one is the student network.
4) *Selective Attention:* Selective attention [156]–[159] is a technique of targeting the interested points, while ignoring the other irrelevant elements or objects. It is derived from the vision system of human beings [160], [161]. When we stare in a specific direction, we only target one or a few objects at a time, and other regions are blurred out.
5) *Low-Rank Decomposition:* Low-rank decomposition/factorization uses matrix/tensor decomposition to estimate the informative parameters. A weight matrix $A$ with $m \times n$ dimension and having a rank $r$ is replaced by smaller dimension matrices. This technique [162]–[166] helps by factorizing a large matrix into smaller matrices. Recently, the tensor-wise decomposition techniques [167]–[170] are prevailing and frequently adopted in model compression of deep CNNs.
6) *Bits Precision:* For bits precision, the number of bits used to represent weight parameters is suppressed for reducing the storage and computation [171]. As an

TABLE I
MAIN STREAM MODEL COMPRESSION TECHNIQUES WITH THEIR PROS AND CONS

| Techniques | Pros | Cons |
|---|---|---|
| **Pruning** | • Applicable in the process of training and post-training<br>• Applicable to fully connected networks or convolutional networks (layers)<br>• Balance the trade-off between inference time (model size) with accuracy [135] | • It is not as helpful as replacing with a better architecture [135]<br>• The model size benefits cannot lead to a significant benefits on implementation latency, especially in common platforms (e.g., TensorFlow) |
| **Quantization** | • Applicable in the process of training and post-training<br>• Applicable to fully connected networks or convolutional networks (layers) | • The convergence of the compressed model is affected. Learning rate is required to be small to ensure a good performance of the networks [136]<br>• Since the gradient cannot propagate back through discrete neurons, the traditional back-propagation training is infeasible. Thus, approximation methods are preferred to estimating the gradients of the loss function instead [136] |
| **Knowledge Distillation** | • The pre-trained teacher network makes the student network easier and faster to train (less training data and smaller size of the model required)<br>• Can reduce the size of a network regardless of the structural difference between the teacher and the student models | • The student model may require a larger dataset with a longer training process to train without a pre-trained teacher model |
| **Selective Attention** | • Faster inference<br>• Smaller model (e.g. a face detector and cropper could be only 44 KB)<br>• Accuracy gain (by focusing downstream AI on only the regions/objects of interest) | • Supports only training from scratch |
| **Low-rank Factorization** | • Applicable in the process of training and post-training<br>• Applicable to fully connected networks or convolutional networks (layers)<br>• When applied in the process of training, it can reduce training time | • Computationally expensive<br>• Cannot perform global parameters compression<br>• Factorization requires extensive model retraining to achieve convergence |
| **Bits Precision** | • Float-to-integer transferring simultaneously reduces the size of storage as well as computational cost<br>• It is flexible to use any number of bits to perform the DNN operations<br>• Binarization techniques can achieve a high-order compression without much performance degradation | • It is time-consuming to select an optimal number of bits for the specific estimation<br>• Complexity from float to integer along with a higher accuracy compromise |
| **Transfer Learning** | • A small size of data set is adequate for the training process of the transferred domain<br>• No more training from scratch saves a lot of time and computation resource | • The target domain needs to be similar with the initial domain<br>• Can not remove layers with confidence to reduce the number of parameters |
| **Early Exit** | • Applicable during and after training<br>• Availability of multiple exit points depending on the goal of tasks<br>• The early exit model can integrate with other classifiers (e.g., SVM) | • The exit point needs to be carefully chosen to avoid a sharp drop of performance [137] |

example, we take 32 bits to store the weight parameters in a matrix of the DNN model. It can be further compressed to 8 bits by replacing floats with integers. This transformation from float to integer simultaneously reduces storage and computation requirements. However, the complexity might increase during the conversion to achieve higher accuracy. The existing researches on bits precision [133], [172]–[177] are devoted to addressing these issues.

7) *Transfer Learning:* Transfer learning [178], [179] is another branch of ML technique, where a model designed for a task is reused as the starting point for models on the other tasks. It is a prevailing method in DL where pretrained models are used as the starting point on computer vision and NLP tasks given the vast compute and time resources required to develop neural network models on these problems, where the pretrained model then can be used to transfer to other problems without training from the scratch which can be regarded as a form of model compression [156], [180], [181]. Specifically, the most common strategy is to leverage the transferred/compact convolutional filters, where special structural convolutional filters are designed to reduce the parameter space and save storage/computation [181]–[183].

8) *Early Exit:* While DNNs benefit from a vast number of layers, it is often the case that a large number of samples can be classified accurately with much less computation. The related works have been proposed to leverage the idea of early exiting before the predefined endpoint of the pipelines. Panda *et al.* [184] observed that a large number of samples can be classified easily and require less processing than some more difficult samples and they obtain this in terms of energy savings.

Teerapittayanon *et al.* [185] investigated a selective approach to exit placement and criteria for exiting early. Recently, similar works [137], [186], [187] also intended to reduce the model size via early exit technique in a wide range of applications.

The best part is, all of the above techniques are complementary to each other. They can be applied as is or combined with one or multiple techniques. By using a three-stage pipeline, pruning, quantization, and bits precision to reduce the size of the pretrained model, where the VGG16 model trained on the ImageNet data set was reduced from 550 to 11.3 MB [188], which saves a huge amount of time in the training phase. Most of the techniques discussed above can be applied to pretrained models, as a post-processing step to reduce the model size and increase inference speed. In addition, they can be applied during training as well. Thus, model compression could be a vital basis for accomplishing the real-timeness in modern IoT systems since it provides a wide range of time-saving techniques to meet the deadline requirement without much performance degradation. The next step is naturally about how we implement such model compression techniques in a real-time pipeline. The practical real-time IoT systems typically have complicated scenarios which consist of multiple tasks (e.g., DL applications) with periodic/sporadic constraints, where the simple case-by-case model compression is no longer valid and is urgent to be adapted.

*2) Real-Time Pipeline:* Implementing the aforementioned model compression techniques into real-time IoT systems is challenging, since most systems are multiprocess and ML models run on shared resources, where the system-wise latency and accuracy cannot be guaranteed. To better analyze the bottleneck when we deploy ML or DL in real-time IoT systems, we summarize the mainstream pipeline [189]–[191] into three steps in sequential, which are listed as follows.

1) Profiling ML models.
2) Selecting proper ML model for each task.
3) Scheduling ML tasks in a real-time manner.

Before we review each of the steps in detail, we take a brief look at the classical two-phase procedure in ML especially for DL (DNN), which are *Training* and *Inference*. Training is a procedure to guide a (deep) neural network to perform the desired task (i.e., object recognition or the next word prediction in a sentence) by feeding the data in it, conducting a trained model for further use. In the training phase, the model predicts the representation of each data sample based on the labels. The prediction error then feedbacks to update the power of connections between the neurons. Along with the training process, such connections are continuously adjusted until the model achieves a satisfying level of prediction accuracy or it cannot get better anymore.

As shown in Fig. 6, the researcher has prepared a set of training data containing hundreds of images (e.g., a person, a bicycle, or a strawberry is labeled for each image). In the training phase, the model (DNN) makes a prediction on the images fed to it. Specifically, in the upper training phase of Fig. 6, the model misclassifies an image as a strawberry which has a ground-truth label of a bicycle. This error feedbacks through a so-called backpropagation, where the weights are adjusted to

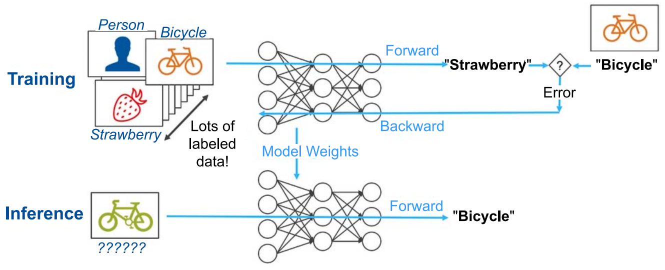


Fig. 6. Training and inference process of DNN [192].

mitigate the error so that the same image will not generate the wrong label (with a higher probability to generate the correct label) in the future predictions. Such a training phase continues (i.e., feeding images and updating the weights according to the possible errors) until the predefined training steps are executed or the desired prediction accuracy is achieved. In this moment, the model can be regarded as a trained one and is ready for future prediction tasks on those unseen images (never fed into the training phase). Note that the training phase usually consumes a huge amount of computation resource especially for those large and complex DNN models. Andrew Ng, who is the former chief scientist at Baidu's Silicon Valley Lab, says training one of Baidu's Chinese speech recognition models requires not only four terabytes of training data but also 20 exaflops of computing—that is 20 billion math operations—across the entire training cycle. Thus, the current training is rarely considered to be real-timely scheduled, while we still explore the potentials and summarize some possible solutions in Section V-B2.

Once the training phase is finished, the trained model is then used to make predictions on the unseen data, which is the so-called inference phase in the learning process. As aforementioned, the training phase involves inference actually since the forward propagation can be regarded as the classify the input images for weights updating. In this case, deploying a trained DNN for inference can be trivial, where usually a simple forward propagation of the trained model is enough for a standard inference phase. Due to this fact, the researchers prefer to make efforts to improve or optimize inference of DNN in real-time IoT systems. In the rest of this section, we summarize the previous works on the inference process of ML/DNN in three steps to achieve the real-time pipeline.

*a) Model profiling:* The *first* step is to construct a trustworthy and comprehensive profile of ML models in terms of execution time, performance (e.g., accuracy), and computation resource cost. Unless we are accurately informed of these characteristics, we are barely able to control the timing of running ML models to solve complex problems in multitask scenarios. For the aforementioned three kinds of ML algorithms, neural network-based (deep) learning is one of the most adaptable and scalable algorithms due to its highly modularized (e.g., stacked and replaceable layered design) architecture. Although the training process is uncertain and time consuming, once we have the well-trained model, the time consumption on inference is with small perturbation in an acceptable range, where we can estimate the relatively accurate execution time based on the architecture design of specific DNNs.

Characterizing DL model inference is complex as its performance depends on the interplay between different levels of the HW/SW stack, e.g., frameworks, system libraries, and hardware platforms. A model inference pipeline can be described as a top-down nested flow. At the top, a model-level evaluation pipeline plays an important role. Three components are included in this level, which are input preprocessing, model prediction, and output post-processing. Inside the level of model prediction, the layer-level components, e.g., convolution layer, batch normalization layer, softmax layer, etc. Further within each layer are the GPU kernel-level components, a sequence of CUDA API calls or GPU kernels invoked by the layer. It is critical to obtain a holistic view of the execution to identify and locate performance bottlenecks due to the complexities of model inference.

Traditional profiling strategies bring a partial view of model execution. For example in ML profiling on GPUs, in order to measure the latency of model level, we can insert timing code in the model prediction step of the inference pipeline. To capture the layer-level information, we can leverage the ML framework's profiling capabilities [193], [194]. Then, to capture GPU kernel information, we use GPU profilers such as NVIDIA's nvprof [195] or Nsight [196]. To correlate profiled events with model layers, Li *et al.* [197] proposed an across-stack profiling design named XSP which leverages distributed tracing to aggregate and correlate the profiles from different sources into a single timeline trace. Through the leveled experimentation methodology, XSP copes with the profiling overhead and accurately captures the profiles at each HW/SW stack level.

Apart from software and hardware profiling, DNN inference model offloading to Cloud or Edge network servers is becoming more and more practical with the increasing importance of Edge Computing. Related work [198] demonstrates that the current approach of DNN inference profiling without considering the dynamic system load of the edge device results in suboptimal partitioning of the DNN algorithm and provides a basic solution approach to that. To further mitigate the DNN profiling challenges in edge computing, another group of researchers built a framework—Edgent [199]—a collaborative and on-demand DNN co-inference framework with device-edge synergy. Edgent has two advantages: 1) adaptive DNN partitioning design benefits the profiling between device and edge for the purpose of coordinating the powerful cloud resource and the proximal edge resource for real-time DNN inference and 2) DNN right-sizing that further reduces computing latency via early exiting inference at an appropriate intermediate DNN layer. In addition, considering the potential network fluctuation in real-world deployment, Edgent is properly designed to specialize in both static and dynamic network environments.

*b) Model selection:* Once we have acknowledged the characteristics of a wide range of DNN models, the *second* step is to match suitable DNN models for specific tasks to meet the deadline requirement. The goal of such optimization is typically minimizing the resource consumption and maximizing the overall model performance (e.g., accuracy in image classification) simultaneously. However, it is not straightforward to process DNN-based workloads in real-time IoT systems equipped with GPU-accelerated platforms, due to the need to satisfy two (usually) conflicting goals: timing predictability and resource efficiency. Timing predictability (i.e., meeting deadline requirement) is one of the most important factors in the certification required for some time-critical systems, e.g., autonomous driving systems. Temporal correctness is crucial to the functional correctness of an autonomous car (e.g., accomplishing the task of object detection within a strict latency to signal automatic brake systems). On the other hand, autonomous cars need low-power consumption, due to their strict size, weight, and power (SWaP) requirements. Unfortunately, timing predictability and resource efficiency are usually in conflict. It is reasonable that the former requires reserving sufficient resources for guaranteeing latency even in the worst case; while the latter often desires to allocate just enough resource that barely meets the needs of the current job.

Two branches of efforts have been separately spared either from efficient computation and anytime prediction. Many prior studies propose computationally efficient variants of traditional machine-learning models [200]–[207]. Most of these studies focus on how to incorporate the computational requirements of particular computing features in the training of machine-learning models such as (gradient-boosted) decision trees. Apart from these explorations in the training of statistical and shallow ML models, FractalNets [208] performs anytime prediction by progressively evaluating subnetworks of the full network and deep architecture. FractalNets are not explicitly optimized for computation efficiency and with in static strategy. Rather than static strategy, [209]–[211] are proposed to reduce the batch computational cost and adaptively evaluate neural networks. Specifically, the adaptive computation time method [209] and its extension [210] perform an adaptive evaluation on test examples to save batch computational cost and focus on skipping units rather than layers. In [212], a "composer" model is trained to construct the evaluation network from a set of submodules for each test example (can be regarded as a different task under assignment). The Feedback Networks [213] enable early predictions by making predictions in a recurrent fashion, which shares parameters among classifiers.

On top of these methods, [191] adopts a specially designed network with multiple classifiers, which are jointly optimized during training and can directly output confidence scores to control the evaluation process for each test example. Note that this work uses a single CNN with multiple intermediate classifiers that is trained end to end. Unfortunately, the aforementioned methods cannot guarantee a relatively strict real-time manner without exception. Although the characteristic of "anytime" can improve the efficiency and flexibility of DL inference tasks (some take care of both training and inference phase), the system-level objectives are far from being accomplished since a tradeoff between resource efficiency and timing predictability should be optimized. In other words, multiple tasks will compete for the limited resources (with deadlines) and the real-time IoT system intends to maximize the overall performance. How to organize Ml/DL workloads (tasks) in a real-time manner becomes urgent and critical, where it is natural to solve this issue in the next step to treat it as a real-time scheduling problem.

*c) Model scheduling:* For the *third* step, previous works focus on scheduling the ML tasks to fulfill the “conflict” objective. For general workloads, several recent works have been done on optimizing latency and energy simultaneously in multicore systems [214]–[217]. For example, in autonomous driving systems, object detection plays a vital role in different operations, so the object detection needs to be real-time obviously. Hence, an end-to-end object-detection analysis for real-time applications is imperative. In [218] and [219], the execution timing behavior of end-to-end object detection of autonomous-driving systems has been analyzed. Based on timing analysis, Jang *et al.* [219] proposed three optimization techniques: 1) on-demand resource allocation for capture; 2) zero-slack pipeline; and 3) contention-free pipeline. Using the zero-slack optimization technique, it is possible to completely avoid the detection mechanism’s queuing delay. As we investigated, there are not so many works targeting DL/DNN workloads and exploring their unique characteristics in performance optimization. A particular aspect of DNNs is that they have multiple layers, each of which is with different complexity of computation and a variety of features. Bateni and Liu [189] and Bateni *et al.* [190] demonstrated that the power consumption pattern differs dramatically among different layers and with varying system configurations. The mainstream layer-oblivious energy/latency optimization algorithms [220], [221] that focus on the general workload may not be suitable to DNN-based automobiles since they do not explore and exploit per-layer characteristics.

Moreover, most existing algorithms consider energy optimization in environments with soft timing constraints [222]–[224]. They improve latency only on a best effort basis but clearly cannot obtain the required timing predictability. The resulting state of affairs is rather unsettling: the revolution of DNN is enabling dramatically better autonomy and services in the automobile, but the required timing predictability and energy efficiency cannot be achieved simultaneously in any DNN-based automobile system. To handle the rigorous timing constraints, [189] and [190] propose a timing-predictable runtime system that guarantees hard deadlines of DNN workloads via efficient approximation. They design a layer-aware subdeadline assignment policy to include approximation potential, a measurement of the effectiveness of approximation on a per-layer basis. Subsequently, they propose a formulation that can assign individual approximation requirements to layers based on their approximation potential. This formulation takes into consideration the real-world limitations of DNN approximation. As described in [190], one major drawback of naively enabling concurrency through resource sharing is the fact that different DNNs might hardly overlap since their cumulative resource usage exceeds the capacity limits of GPU. A trending direction for modern real-time pipeline for DNN is to develop a runtime solution since the fact that different approximated configurations of a DNN have different resource utilization profiles. Bateni and Liu [189] designed and implemented a runtime system for enhanced runtime multitasking performance, which exploits the mutually supplementary relation between approximation and resource sharing.

Although [189] and [190] provide a complete runtime solution for a real-time pipeline, it still faces the unschedulable issue since the dynamic mechanism itself has a limitation when upcoming tasks (DNN workload) exceed its predefined threshold (e.g., the next task can never be scheduled based on the current system status). Thus, such a scenario leaves a future direction to design a clairvoyance system that can not only statically preschedule (i.e., check with schedulability tests and generate an optimal schedule before the execution) the ML workload based on the plan given by the user but also is able to dynamically adapt to some interruption or insertion of some extra tasks.

### C. Privacy and Security in Real-Time IoT Systems

The security and privacy issues have been seriously taken care of in most modern IoT systems. There exists a wide range of studies, including at least four major dimensions suggested by [225]. For the first dimension, the author mentions the limitation of implementing security in devices (e.g., computing resource, battery, and scheduler) of IoT systems with the related solutions (e.g., encryption technologies). The second one is the type of IoT attacks, where the possible attacks could be physical, remote, local, etc. Then, the third one is the mechanisms and architectures designed and implemented for authentication and authorization. The last one is layer-wise security issues, such as physical and network. While ML/DL actually plays important role in all the dimensions.

IoT devices that are more easily compromised compared to desktop computers have led to a rise in IoT botnet attacks. In order to mitigate this threat, Meidan *et al.* [226] have proposed the use of deep autoencoders (DAEs) to detect anomalous network traffic from compromised IoT devices. DL with its capabilities, such as high-level feature extraction capability, self-taught, and compression capabilities make it an ideal hidden pattern discovery that aids in discriminating attacks from benign traffic. Therefore, study [227] proposes a DL approach based on stochastic gradient descent (SGD), which enables the detection of attacks in the social IoT. Besides, Roy and Cheung [228] have proposed a DL technique that enables intrusion detection in IoT networks using the bidirectional LSTM RNN (BLSTM RNN). Furthermore, Haddadpajouh *et al.* [229] have proposed a DL model using LSTM to detect malware in IoT based on OpCodes sequence. We also showcase a series of representative works applying ML/DL in IoT systems which is across multiple dimensions. Dawoud *et al.* [230] have introduced a framework for IoT based on software-defined networking (SDN). Zhou *et al.* [231] have discussed that IoT applications face major security issues in confidentiality, integrity, privacy, and availability. Brun *et al.* [232] have proposed a DL approach with dense random neural networks (DRNNs) to predict the probability of an ongoing network attack based on the packet capture. Study [233] proposes a model that uses LSTM and CNN to distinguish ransomware and goodware in networks.

However, none of the above studies has involved real time as a constraint to applying ML/DL in IoT systems. Compared to general-purpose computing and networked systems, privacy and security used to be less concerned with the research communities of real-time embedded systems, as embedded systems were assumed to be isolated from an open or adversarial environment [234], [235]. However, in IoT settings [50], [51], where devices are interconnected with interoperations, privacy and security have become critical issues of real-time IoT systems with ML and data analytic components [236], [237]. Generally, we categorize the existing works into two folders as follows.

1) *Privacy and Security Enhancements for Data Processing and Machine Learning Over IoT:* As IoT devices, such as healthcare IoT for medical purposes, usually have to aggregate and process information related to human subjects [238], [239], there frequently needs to address the privacy and security issues for ML or data processing algorithms on these devices. As early as [240], privacy preservation in mobile crowdsourced sensor networks has been studied, where the authors proposed to leverage anonymous participants to protect the location privacy of mobile users. Later, to further lower the privacy and security risk introduced by the data aggregation procedure [241] in distributed IoT, aggregation-free distributed sensing has been proposed in [242] with mobile sensors, where the authors proposed using decentralized SGD with multiparty computation [243] to enhance the compressive crowdsensing algorithms [244] for spatial–temporal monitoring. Besides data aggregation, ML for statistical supervised learning has been improved and secured for distributed IoT applications in [243] and [245]. Zhang *et al.* [246] proposed DeepPar—a privacy preserving and asynchronous DL for Industrial IoT over distributed data sets. For similar purposes, Li *et al.* [247] proposed SmartPC that secures the privacy of distributed data sets in a federated learning framework while minimizing the overall energy consumption on nodes. Rather than the separation of samples over distributed data sets, features of the same group of samples might be split and stored in different nodes. Feng *et al.* [248] studied a novel secure gradient boosting machines model (SecureGBM) to enable federated learning in such settings. In addition to tackling the privacy and security issues in a distributed manner, data federation with trusted execution environments (TEEs) [249]–[252] is yet another way to perform data aggregation and ML using trustworthy infrastructures.
2) *Privacy and Security Enhancements for IoT Systems Using Data Analytics and Machine Learning:* In addition to securing the privacy and security of ML tasks over IoT systems, ML techniques could also enhance the privacy and security of real-time IoT. Specifically, Shakeel *et al.* [253] proposed to use learning-based Deep-Q-Networks to enhance the security and privacy in IoT-based healthcare systems. The work [254] tried to enhance IoT security through automatic authentication of wireless nodes using *In-Situ* ML algorithms. Roopak *et al.* [255] reviewed DNNs used in IoT cyber security. Zolanvari *et al.* [256] studied the effects of imbalanced data sets on IoT security with ML. The work [257] studied ML algorithms to classify the risk of IoT security issues. Sagduyu *et al.* [258] studied the use of adversarial learning algorithms to promote the security of IoT. More work could be found in the following surveys and technical reviews [259]–[264].

In addition to the above privacy and security issues for applications of Real-time IoT with ML, some recent works have demonstrated the possibility to attack real-time schedulers for IoT through reversing the execution times and orders of tasks [102], [265], [266] using ML techniques, where the final goal of these attacks is to interfere the schedule and execution of mission-critical tasks and make system failures. A possible way to fight against these attacks is to incorporate random reshuffling in scheduling [102]. For one of the most popular branches of ML-based privacy enhancements, federated learning plays a more important role in modern real-time IoT systems. To construct a reliable and sustainable IoT system, a series of works study federated learning with respect to different privacy protection mechanisms. Lim *et al.* [267] and Kang *et al.* [268] intended to design reasonable incentive mechanisms to improve the real-time cooperation among the learning agents. Another group of works [269]–[273] explores the edge-empowered mechanisms, such as Blockchain-based edge learning and network virtualization to strengthen the security in the federated learning process. However, the real-time characteristics and high-standard privacy and security protection are rarely balanced in the current state-of-the-art federated learning strategies. Thus, it is still remained to be explored further in future studies.

## IV. Applicability of Machine Learning in Real-Time IoT Systems

In this section, we try to summarize the applicability of ML techniques in modern real-time IoT systems in terms of industrial practice.

First, we categorize the real-time IoT systems/applications in main aspects and briefly introduce several industrial problems with solutions which are divided into the traditional pipelines and the ML-based techniques. Then, the real-time characteristics are checked for each ML-based solution. As shown in Table II, we pick up six representative real-time IoT systems/applications, including utilities, manufacturing, healthcare, insurance, retailing, and transportation. The specific problems and solutions are described as follows.

1) In utilities, we are eager to save energy by predicting the usage and dynamically allocation. The traditional way is to analyze the meters for demand–supply prediction via statistical tools, which is lagging and does not have a real-time guarantee. But with those ML analyzers to the gas, electronics, and water, we can store the well-trained model on the server, predict the trend of usage on the fly, and dynamically adjust the model. Furthermore, we can make load balancing and dynamical allocating.
2) In manufacturing, there are lots of human resources that can be saved by a real-time IoT system with cameras and controllers. The system detects abnormal operation, alerts, and operates actions accordingly, which can save lots of resources preventing the fault by predicting it. The main differences between traditional and ML-based solutions are a) the prediction accuracy increases by ML

TABLE II
APPLICABILITY OF ML TECHNIQUES IN REAL-TIME IoT SYSTEMS/APPLICATIONS

| Industries | Problems | Facilities | Solutions | Applicability |
|---|---|---|---|---|
| **Utilities** (energy, water, gas, and etc.) | • Real-time collection of usage data<br>• Demand-supply prediction<br>• Load balancing<br>• Dynamic tariff generation | • Sensors and meters for energy, gas, and water etc. | • Traditional: Historical usage analysis, usage prediction, demand-supply prediction via statistical tools (e.g., linear regression, SVM [274])<br>• ML/DL-based: real-time utility prediction and management based on ML/DL [275]–[277]. | • Applicable in real-time manner. |
| **Manufacturing** | • Remote monitoring and diagnostics in case of failures<br>• Production line automation | • Supervisory control and data acquisition systems (SCADA) [278]<br>• Programmable logic controllers (PLCs) [279]<br>• Cameras<br>• IoT devices mounted or embedded | • Traditional: Anomaly detection via statistical strategies (e.g., chi-squrare [280], divide and conquer [281]).<br>• ML/DL-based: Anomaly detection and automatic quality monitoring using ML/DL [282]–[284]. | • Applicable in real-time manner. |
| **Healthcare** | • Remote expert/doctor consultation/monitoring<br>• Chronic disease management<br>• Elderly care<br>• Wellness and fitness programs | • Wearable and personal medical devices<br>• Smart mobile phones | • Traditional: Historical correlation analysis via statistical tools [285]–[287].<br>• ML/DL-based: Anomaly detection in recorded medical data via ML/DL [288]–[291]. | • Applicable in real-time manner. |
| **Insurance** | • User data collection (e.g., condition of home devices for home insurance, driving habits for car insurance)<br>• Prediction of property damage or rate of depreciation<br>• Remote inspection and assessment of damage and accidents | • Sensors that depict the condition/usage of the insured entity | • Traditional: Usage pattern detection via statistical tools [292]–[294].<br>• ML/DL-based: Anomaly detection and automated assessment via ML/DL [295]–[298]. | • Applicable in real-time manner. |
| **Retailing** | • Real-time knowledge of the customers' context/profile (e.g., presence, location, preference, and so on)<br>• Monitoring supply chain inventory | • Sensors that can capture end-user and inventory context (e.g., RFID, locations sensors, robots with sensors, specialized devices) | • Traditional: Analytic to extract context from raw data by statistical tools [299]–[301]<br>• ML/DL-based: Context-aided real-time user profiling via ML/DL [302]–[305] | • Applicable in real-time manner. |
| **Transportation** | • Real-time vehicle tracing and optimization for logistics and public transportation systems<br>• Asset management and tracking | • On-board vehicle gateway devices<br>• RFID tags<br>• Sensors | • Traditional: Real-time alert to driver/operator, dashboards/ control panels in command and control centers using active infrared illuminator and software implementation [306]–[309]<br>• ML/DL-based: Visualization, prediction, optimization, and decision support systems for associated transportation systems via ML/DL [310]–[312] | • Applicable in real-time manner. |

solutions and b) the automation is drastically improved by ML solutions in the manufacturing management.

3) In healthcare, the issue is personalized health history tracking. If the patient has several wearable devices that track those data, doctors can have a more accurate analysis of the patient. And this is far cheaper to track everyone's health condition than hiring a personal nurse. In terms of data analysis from wearable devices, ML solutions are more intelligent than traditional statistical tools, where the DL models are more suitable for high-dimensional data and meantime fulfill the real-time constraints.
4) In insurance, the industry analyzes the property in the financial papers. But recently, we can leverage the data integration from personal devices. By collecting and analyzing those data, we can wisely customize personal insurance that fits personal situations. Risk estimation is crucial and can be regarded as a kind of anomaly detection. Through comprehensive investigation, the ML solutions demonstrate superiority in terms of real-time and efficacy.
5) In retailing, we intend to predict when will our customer be, what he/she wants to buy and how much he/she will purchase. The sensors can be placed in the store and warehouse, and the data can be gathered from the Internet, such as shopping apps and Web stores. As we know, to control the cost of logistics in the supply chain, we need to preallocate specific goods ahead of the peak season, which could be efficiently investigated by ML solutions based on the profiles of customers/users. Furthermore, fast analysis and reaction also play important roles in supply chain management, which require inference in a real-time manner. However, the traditional solutions are inefficient to extract the context so that it is hard to achieve the hard inference deadlines with satisfied prediction accuracy.
6) In transportation, analysis on the flow data of human beings as well as the vehicles is the first step to commit the management of transportation. For example, if the supply of vehicles meets the need for transportation, it's an efficient allocation to the public transportation timings (e.g., buses timing and subways timing). With the dynamical arrangement of the limited resources, we can lower the cost of operation when idling or enhance the service level when busy working. To this point, ML solutions can provide assistance in a wide range of aspects, including real-time visualization, prediction, optimization, and decision support, which surpass the traditional solutions with simple feedback controls.

Note that we have confirmed the soft/hard[1] guarantee of the real-time manner in each ML/DL-based solution through a comprehensive investigation in literature and selecting several representative research/application studies to present in Table II.

## V. DISCUSSION AND FUTURE RESEARCH

### A. *Toward Machine Learning for Real-Time Systems*

As RTSs have started to be used in various applications, real-time IoT design faces unprecedented challenges.

[1]The soft/hard real time are first defined in [2] and used depending on the type of application.

Accordingly, new research problems arise to tackle those challenges. Here, we will discuss a few challenges and general issues in real-time IoT systems that can be mitigated by leveraging ML algorithms.

1) *Predictability:* Predictability is the expected behavior of RTSs. The predictability of the system using exact analysis becomes impractical with increasing system complexity. Therefore, a probabilistic predictability analysis could be a good alternative to exact analysis for complex RTSs. The MCS design community has already explored probabilistic system behaviors for system mode-switch prediction. Predictability analysis can generally be performed by leveraging ML algorithms on cache/memory or I/O data access patterns and throughput analysis.
2) *Malicious Behavior Detection:* To defend or recover a system under attack, detection of the system's malicious behavior is imperative. Most of the existing work on run-time system monitoring for malicious attack detection is developed for general-purpose systems. There are very few real-time attack detection methods that are too slow to recover the system before the deadline or produce many false alarms. So, there is a gap between two contradictory goals of fast detection and small false-positive. ML-based algorithms can play a vital role in fast malicious attack detection methods with tolerable false-positive results.
3) *Real-Time System Recovery:* Unlike general-purpose computing systems, RTS tasks under attack cannot be shut down to protect malicious activities as one of the goals of the attacker is also to shut down the process. So, it is necessary to develop an attack recovery method to recover the system instead of simply killing the infected process. A real-time attack recovery method using linear approximation is represented in [313]. In the current approach, the tolerance of the recovery method is relatively high, and the system can easily reach an undesirable state with a variation of other system parameters. Therefore, a more secure recovery system is desirable. ML can be used with more research on this perspective.

### B. Toward Real-Time and Schedulable Machine Learning

On the contrary, the deployment of ML or DL workloads in IoT systems could also be further improved in more intelligent and efficient ways with real-time scheduling. In this section, we separately discuss the potential future direction in either the inference phase or training phase of ML, which are two key steps in common learning procedures.

*Inference:* Serving ML (especially for DL) inference in a timely manner is mandatory in a wide range of applications, such as self-driving and traffic monitoring, existing works, which intend to reduce inference time using tiny architectures (e.g., MobileNet [314]) or compressing DNNs, cannot provide any real-time performance guarantee [1]. In addition to the real-time performance, high inference accuracy is also required in these applications. However, larger DNN models with more parameters and consuming longer inference time usually deliver higher testing accuracy [315]. Thus, a nontrivial tradeoff between inference complexity and accuracy of DNNs is desired for real-time DNN inference serving systems design.

In order to balance inference time and accuracy in the design of the timed systems, the researchers have proposed numerous solutions from the neural networks, and schedulability aspects. For example, multiscale DenseNet (MSDNet) [315] and the approximation-aware network (APNET) [189], which we have summarized in the previous sections. In addition to networks, DNNs inference could also be accelerated in the massive online systems through resource scheduling [316]–[318]. For example, DART [319] studied to schedule inference tasks for multiple DNNs on CPU/GPU. PACE [320] proposed preemptive scheduling algorithms to expedite distributed DNN training. PREMA [321] proposed preemptive neural processing units for real-time scheduling of DNNs inference tasks. All these efforts intend to design timed systems for DNN inference and use preemptive/nonpreemptive scheduling techniques to schedule DNN inference/training tasks over shared resources (e.g., GPU). However, they all failed to provide schedulability tests [1] to verify the design of DNNs inference serving systems or approximate optimal tradeoff between accuracy and inference time under schedulability constraints [1]. Thus, one of the urgent directions is to design RTSs that include the following characteristics,

1) The systems can follow the "early-exit"/"subnetwork" strategies to handle the DNNs inference with time–accuracy tradeoff.
2) The systems are suggested to treat the scheduling (multitask) problem as a constrained optimization problem, with overall expected inference accuracy as the objective and utilization-based schedulability tests as constraints.
3) To solve the constrained optimization problem (which is probably NP-hard), the systems require the solvers, which can approximate the optimal design of the RTS with multiple DNNs inference tasks, under specific scheduling algorithms (e.g., EDF and rate monotonic scheduling (RMS) algorithms).

*Training:* Beyond the inference task for ML applications, the training process is also of crucial importance and works as a resource-dominant part of ML applications. In order to improve the efficiency and effectiveness of resource management in the whole of an ML application, the traditional routine is to set a fixed off-chip training model and feed the well-trained model to a speed-up inference chip (e.g., NNP-I [322]), which means the resource efficiency is mainly optimized in the inference and its correlated connection part (with a well-trained model). This brings the following concerns and challenges in the resource-constrained settings.

1) *The Lack of Adaptability in the Inference Model on the Chip:* The training model is typically fixed for specific ML tasks, and need to be replaced (redesign the architecture) and then retrained when a new task arrives (e.g., transfer face detection to car detection), where these modifies in the training process is actually resource intensive. Thus, along with the limited resource for the training process, the adaptability of the inference model is lacking indeed.
2) *The Computation-Intensive Backpropagation Issues:* Fully training a DNN for a specific application usually

consumes a large amount of time and space resources due to the process of backpropagation, where actually some of the weight updating and layer bypassing are redundant and trivially influence the final result to some degree. Especially for the convolutional layers which are not highly dependent on a specific application (e.g., image recognition), it is a waste of recalculation for the backpropagation of the layers when the target application just changes a little (e.g., from cat images to dog images classification).

3) *The Emerging Need of Real-Time Capability and Scalability for the Resource-Constrained Environment:* For example, when we apply an image recognition application on the mobile devices, we need to consider the tradeoff between the limited battery capacity and the performance of the recognition task (can include the accuracy of the image classification, the resolution which can be identified and also the real-time deadlines needed to be satisfied). Based on the fixed design, the traditional DL network system is not scalable with the complicated scenario of the practical resource-constrained environment.

To address the above urgent concerns, a possible adaptive multiscale multitask real-time recourse-constrained neural network system is in need. The key point is to design a (re)learnable neural network system which can be adapted to varieties of tasks and different levels of constrained resources. Ideally, given the current resource capacity and the real-time deadline guarantees, the system should automatically provide efficient models with their performance intervals (e.g., accuracy with specific deviation) and time consumption. In this way, the user can decide to either pursue higher accuracy or faster computing under current settings. Instead of the traditional design that separates the training and inference process, the new real-time neural IoT system should integrate these two parts. Thus, on top of the whole ML applications, the resource can be managed in a more intelligent and efficient way.

Inspired by the recent works [191], [314], [323], a possible solution can be formed using the early-exit mechanism and the intermediate classifier. The combination of early-exit layers and the statistical classifier (e.g., SVM and random forest) can provide fast results without losing much accuracy compared to a complete CNN or ResNet. In addition, such a combination will differ from the previous multiscale dense network [191], the early-exit layers will be pretrained in a multitask way, where it can obtain the best feature embeddings of more than one task in only one submodel. With these early-exit layers, the system can train the submodel and conduct the inference based on the limited resource while adapting the multitask requirement. Note that the efficient computing in the training and inference part can be controlled by combinatorial optimization to achieve the goal of real-time guarantees. At the same time, the deadlines can be met by carrying out an appropriate degradation of the accuracy in the final results. Moreover, this future direction has the potential to be a promising solution as a design of the next-generation AI neural chip, where it maximally combines the training and inference process of the neural networks on a single chip and naturally increases the adaptability and resource efficiency of the chip.

## VI. Conclusion

In this article, we introduced a survey on the state-of-the-art ML algorithms employed in real-time IoT and embedded systems. Our survey presented the challenges of implementing machine algorithms, real-time IoT system design goals achievable through ML, and the potential research gap for accomplishing the goals. The survey was directed to three broad research problems related to the real-time IoT and embedded systems—addressing the adaptation of ML algorithms, ML algorithms for scheduling problems, and security and privacy issues related to the implementation of machine algorithms. We attempted to summarize all existing ML papers on real-time applications discussing the proposed approaches' strengths and weaknesses. Then, we suggested and/or presented research gaps in the current papers, if any.

We believe ML and artificial intelligence would enormously impact real-time IoT system designs. Although there are significant efforts from the research community and system designers, enabling AI-friendly real-time IoT is still challenging. Therefore, we presented a section on open problems and challenges that are yet to be explored.

## References

[1] J. Liu, *Real-Time Systems*. Englewood Cliffs, NJ, USA: Prentice Hall, 2000. [Online]. Available: https://books.google.com/books?id=855QAAAAMAAJ

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[3] K. W. Tindell, A. Burns, and A. J. Wellings, "Allocating hard real-time tasks: An NP-hard problem made easy," *Real Time Syst.*, vol. 4, no. 2, pp. 145–165, 1992.

[4] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[5] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, vol. 24. Cham, Switzerland: Springer, 2011.

[6] M. Stigge and W. Yi, "Graph-based models for real-time workload: A survey," *Real Time Syst.*, vol. 51, no. 5, pp. 602–636, 2015.

[7] S. Kato and Y. Ishikawa, "Gang edf scheduling of parallel task systems," in *Proc. 30th IEEE Real-Time Syst. Symp.*, 2009, pp. 459–468.

[8] N. C. Audsley, A. Burns, M. Richardson, and A. Wellings, *Deadline Monotonic Scheduling*, Univ. York, York, U.K., 1990.

[9] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. IEEE Int. Real-Time Syst. Symp. (RTSS)*, 2007, pp. 239–243.

[10] A. Burns and R. Davis, "Mixed criticality systems—A review," Dept. Comput. Sci., Univ. York, York, U.K., Rep., 2013.

[11] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, Princeton, NJ, USA: Citeseer, 1994.

[12] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *Proc. Sci. Inf. Conf.*, 2014, pp. 372–378.

[13] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey," *Digit. Commun. Netw.*, vol. 4, no. 3, pp. 161–175, 2018.

[14] M. W. Libbrecht and W. S. Noble, "Machine learning applications in genetics and genomics," *Nat. Rev. Genet.*, vol. 16, no. 6, pp. 321–332, 2015.

[15] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine Learning in Computer Vision*, vol. 29. Cham, Switzerland: Springer, 2005.

[16] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.

[17] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The Elements of Statistical Learning*. Cham, Switzerland: Springer, 2009, pp. 9–41.

[18] H. B. Barlow, “Unsupervised learning,” *Neural Comput.*, vol. 1, no. 3, pp. 295–311, 1989.

[19] T. Hastie, R. Tibshirani, and J. Friedman, “Unsupervised learning,” in *The Elements of Statistical Learning*. Cham, Switzerland: Springer, pp. 485–585, 2009.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

[21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3207–3214.

[22] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Proc. 25th Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 24, 2011, pp. 2546–2554.

[23] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Perform. Eval.*, vol. 64, nos. 9–12, pp. 1194–1213, 2007.

[24] S. Lee and S. Nirjon, “SubFlow: A dynamic induced-subgraph strategy toward real-time DNN inference and training,” in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2020, pp. 15–29.

[25] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[26] Y. Bengio and Y. LeCun, “Scaling learning algorithms towards AI,” in *Large-Scale Kernel Machines*, vol. 34. Cambridge, MA, USA: MIT Press, 2007, pp. 1–41.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[28] J. Levinson *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2011, pp. 163–168.

[29] A. Uçar, Y. Demir, and C. Güzeliş, “Object recognition and detection with deep learning for autonomous driving applications,” *Simulation*, vol. 93, no. 9, pp. 759–769, 2017.

[30] S. Buschjager, K.-H. Chen, J.-J. Chen, and K. Morik, “Realization of random forest for real-time evaluation through tree framing,” in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2018, pp. 19–28.

[31] B. Zhou and J. Xu, “An adaptive SVM-based real-time scheduling mechanism and simulation for multiple-load carriers in automobile assembly lines,” *Int. J. Model. Simulat. Sci. Comput.*, vol. 8, no. 4, 2017, Art. no. 1750048.

[32] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, vol. 53. Cham, Switzerland: Springer, 2003.

[33] A. E. Eiben and M. Schoenauer, “Evolutionary computing,” *Inf. Process. Lett.*, vol. 82, no. 1, pp. 1–6, 2002.

[34] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, vol. 65. New York, NY, USA: Wiley, 2005.

[35] H. Kautz and B. Selman, “Pushing the envelope: Planning, propositional logic, and stochastic search,” in *Proc. Nat. Conf. Artif. Intell.*, 1996, pp. 1194–1201.

[36] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[37] M. R. Hestenes, “Multiplier and gradient methods,” *J. Optim. Theory Appl.*, vol. 4, no. 5, pp. 303–320, 1969.

[38] E. Rivlin, M. Rudzsky, R. Goldenberg, U. Bogomolov, and S. Lepchev, “A real-time system for classification of moving objects,” in *Object Recognition Supported by User Interaction for Service Robots*, vol. 3. Manhattan, NY, USA: IEEE, Aug. 2002, pp. 688–691.

[39] T. Radil, P. M. Ramos, F. M. Janeiro, and A. C. Serra, “PQ monitoring system for real-time detection and classification of disturbances in a single-phase power system,” *IEEE Trans. Instrum. Meas.*, vol. 57, no. 8, pp. 1725–1733, Aug. 2008.

[40] Ş. Vădineanu and M. Nasri, “Robust and accurate period inference using regression-based techniques,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 358–370.

[41] B. B. Brandenburg, J. M. Calandrino, and J. H. Anderson, “On the scalability of real-time scheduling algorithms on multicore platforms: A case study,” in *Proc. Real-Time Syst. Symp.*, 2008, pp. 157–169.

[42] A. Predescu, C. Negru, M. Mocanu, and C. Lupu, “Real-time clustering for priority evaluation in a water distribution system,” in *Proc. IEEE Int. Conf. Autom. Qual. Test. Robot. (AQTR)*, 2018, pp. 1–6.

[43] D. Valencia and A. Alimohammad, “A real-time spike sorting system using parallel OSort clustering,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1700–1713, 2019.

[44] S. Makridakis, E. SpilIoTis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward,” *PloS One*, vol. 13, no. 3, 2018, Art. no. e0194889.

[45] C. Molnar, *Interpretable Machine Learning*, Lulu, Abu Dhabi, United Arab Emirates, 2020.

[46] X. Li *et al.*, “Interpretable deep learning: Interpretations, interpretability, trustworthiness, and beyond,” 2021, *arXiv:2103.10689*.

[47] G. Harman and S. Kulkarni, *Reliable Reasoning: Induction and Statistical Learning Theory*. Cambridge, MA, USA: MIT Press, 2012.

[48] L. Sha *et al.*, “Real time scheduling theory: A historical perspective,” *Real Time Syst.*, vol. 28, no. 2, pp. 101–155, 2004.

[49] R. I. Davis and A. Burns, “A survey of hard real-time scheduling for multiprocessor systems,” *ACM Comput. Surveys*, vol. 43, no. 4, pp. 1–44, 2011.

[50] N. Shahid and S. Aneja, “Internet of Things: Vision, application areas and research challenges,” in *Proc. Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, 2017, pp. 583–587.

[51] K. K. Patel and S. M. Patel, “Internet of Things-IoT: Definition, characteristics, architecture, enabling technologies, application & future challenges,” *Int. J. Eng. Sci. Comput.*, vol. 6, no. 5, pp. 6122–6131, 2016.

[52] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, “A survey on application of machine learning for Internet of Things,” *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 8, pp. 1399–1417, 2018.

[53] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[54] T. K. Ho, “Random decision forests,” in *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, vol. 1, 1995, pp. 278–282.

[55] L. Xu and M. I. Jordan, “On convergence properties of the EM algorithm for Gaussian mixtures,” *Neural Comput.*, vol. 8, no. 1, pp. 129–151, 1996.

[56] I. Rish *et al.*, “An empirical study of the Naive bayes classifier,” in *Proc. Workshop Empirical Methods Artif. Intell.*, vol. 3, 2001. pp. 41–46.

[57] G. A. Seber and A. J. Lee, *Linear Regression Analysis*, vol. 329. New York, NY, USA: Wiley, 2012.

[58] D. W. Hosmer, Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.

[59] R. E. Schapire, “A brief introduction to boosting,” in *Proc. IJCAI*, 1999, pp. 1401–1406.

[60] L. Devroye and T. J. Wagner, “8 nearest neighbor methods in discrimination,” in *Handbook of Statistics*, vol. 2. Amsterdam, The Netherlands: North-Holland, 1982, pp. 193–197.

[61] C. J. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[62] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometr. Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[63] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. Cham, Switzerland: Springer, 2013.

[64] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[65] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[66] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, 2014.

[67] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Cham, Switzerland: Springer, 2009.

[68] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition,” in *Competition and Cooperation in Neural Nets*. Cham, Switzerland: Springer, 1982, pp. 267–285.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[70] Y. Shen *et al.*, “CS-CNN: Enabling robust and efficient convolutional neural networks inference for Internet-of-Things applications,” *IEEE Access*, vol. 6, pp. 13439–13448, 2018.

[71] N. Krishnaraj, M. Elhoseny, M. Thenmozhi, M. M. Selim, and K. Shankar, “Deep learning model for real-time image compression in Internet of Underwater Things (IoUT),” *J. Real-Time Image Process.*, vol. 17, no. 6, pp. 2097–2111, 2020.

[72] X. Chen, L. Xie, J. Wu, and Q. Tian, “Cyclic CNN: Image classification with multi-scale and multi-location contexts,” *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7466–7475, May 2021.

[73] Z. Huang, X. Xu, J. Ni, H. Zhu, and C. Wang, “Multimodal representation learning for recommendation in Internet of Things,” *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10675–10685, Dec. 2019.

[74] Y.-J. Choi, Y.-W. Lee, and B.-G. Kim, “Residual-based graph convolutional network for emotion recognition in conversation for smart Internet of Things,” *Big Data*, vol. 9, no. 4, pp. 279–288, 2021.

[75] H. Zhang, Z. Xiao, J. Wang, F. Li, and E. Szczerbicki, "A novel IoT-perceptive human activity recognition (HAR) approach using multihead convolutional attention," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1072–1080, Feb. 2020.

[76] H. Zou, Y. Zhou, J. Yang, H. Jiang, L. Xie, and C. J. Spanos, "DeepSense: Device-free human activity recognition via autoencoder long-term recurrent convolutional network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[77] N. Van Noord and E. Postma, "Learning scale-variant and scale-invariant features for deep image classification," *Pattern Recognit.*, vol. 61, pp. 583–592, Jan. 2017.

[78] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[80] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.

[81] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[82] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[83] P. Soriano, F. Caballero, A. Ollero, and C. A. de T. Aeroespaciales, "RF-based particle filter localization for wildlife tracking by using an UAV," in *Proc. Int. Symp. Robot.*, 2009, pp. 1–7.

[84] R. Glaubius, T. Tidwell, C. Gill, and W. D. Smart, "Real-time scheduling via reinforcement learning," 2012, *arXiv:1203.3481*.

[85] W. Liang, W. Huang, J. Long, K. Zhang, K.-C. Li, and D. Zhang, "Deep reinforcement learning for resource protection and real-time detection in IoT environment," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6392–6401, Jul. 2020.

[86] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 375–388, Jan. 2021.

[87] H. Yang *et al.*, "Intelligent reflecting surface assisted anti-jamming communications: A fast reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1963–1974, Mar. 2021.

[88] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *Proc. 13th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, 2007, pp. 469–475.

[89] Y. Yang *et al.*, "GAN-based semi-supervised learning approach for clinical decision support in health-IoT platform," *IEEE Access*, vol. 7, pp. 8048–8057, 2019.

[90] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning Chapelle, O. et al., Eds.; 2006) [book reviews]," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, p. 542, Mar. 2009.

[91] N. Ghourchian, M. Allegue-Martinez, and D. Precup, "Real-time indoor localization in smart homes using semi-supervised learning," in *Proc. 29th IAAI Conf.*, 2017, pp. 4670–4677.

[92] J. A. Snyman, *Practical Mathematical Optimization*. Cham, Switzerland: Springer, 2005.

[93] C. C. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, Feb. 2006.

[94] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov./Dec. 2020.

[95] D. P. Heyman and M. J. Sobel, *Stochastic Models in Operations Research: Stochastic Optimization*, vol. 2. Mineola, NY, USA: Courier Corp., 2004.

[96] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[97] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with EDF scheduling," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1250–1258, Sep. 2009.

[98] R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1261–1276, Sep. 2008.

[99] M. Joseph and P. Pandya, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, 1986.

[100] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Proc. IEEE 32nd Real-Time Syst. Symp.*, 2011, pp. 34–43.

[101] M. Sjodin and H. Hansson, "Improved response-time analysis calculations," in *Proc. 19th IEEE Real-Time Syst. Symp.*, 1998, pp. 399–408.

[102] M.-K. Yoon, S. Mohan, C.-Y. Chen, and L. Sha, "Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2016, pp. 1–12.

[103] D. Trilla, C. Hernandez, J. Abella, and F. J. Cazorla, "Cache side-channel attacks and time-predictability in high-performance critical real-time systems," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, pp. 1–6.

[104] J. Y. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, 2004.

[105] J. J. Hopfield and D. W. Tank, "'Neural'İ computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, 1985.

[106] T. Ae and R. Aibara, "Programmable real-time scheduler using a neurocomputer," *Real Time Syst.*, vol. 1, no. 4, pp. 351–363, 1990.

[107] C. Cardeira and Z. Mammeri, "Neural networks for multiprocessor real-time scheduling," in *Proc. 6th Euromicro Workshop Real-Time Syst.*, 1994, pp. 59–64.

[108] C. Cardeira and Z. Mammeri, "Preemptive and non-preemptive real-time scheduling based on neural networks," in *Distributed Computer Control Systems*. Pergamon, Turkey: Elsevier, 1995, pp. 67–72.

[109] C. Cardeira and Z. Mammeri, "Neural network versus max-flow algorithms for multiprocessor real-time scheduling," in *Proc. 8th Euromicro Workshop Real-Time Syst.*, 1996, pp. 175–180.

[110] C.-S. Zhang, P.-F. Yan, and T. Chang, "Solving job-shop scheduling problem with priority using neural network," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1991, pp. 1361–1366.

[111] M. P. Silva, C. Cardeira, and Z. Mammeri, "Solving real-time scheduling problems with hopfield-type neural networks," in *Proc. 23rd EUROMICRO Conf. New Front. Inf. Technol.*, 1997, pp. 671–678.

[112] R.-M. Chen, "Reducing network and computation complexities in neural based real-time scheduling scheme," *Appl. Math. Comput.*, vol. 217, no. 13, pp. 6379–6389, 2011.

[113] S. Yang, D. Wang, T. Chai, and G. Kendall, "An improved constraint satisfaction adaptive neural network for job-shop scheduling," *J. Schedul.*, vol. 13, no. 1, pp. 17–38, 2010.

[114] S. Lee, H. Baek, H. Woo, K. G. Shin, and J. Lee, "ML for RT: Priority assignment using machine learning," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2021, pp. 1–13.

[115] Z. Guo and S. K. Baruah, "A neurodynamic approach for real-time scheduling via maximizing piecewise linear utility," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 238–248, Feb. 2016.

[116] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (CAN) refuted," in *Proc. 5th Int. Work. Real-Time Netw. (RTN)*, 2006, pp. 5–10.

[117] F. Cerqueira, F. Stutz, and B. B. Brandenburg, "Prosa: A case for readable mechanized schedulability analysis," in *Proc. Euromicro Conf. Real-Time Syst. (ECRTS)*, 2016, pp. 273–284.

[118] P. Dziurzanski, R. I. Davis, and L. S. Indrusiak, "Synthesizing real-time schedulability tests using evolutionary algorithms: A proof of concept," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2019, pp. 43–55.

[119] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, "Fuzzy dijkstra algorithm for shortest path problem under uncertain environment," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1231–1237, 2012.

[120] K. Agrawal, S. Baruah, Z. Guo, J. Li, and S. Vaidhun, "Hard-real-time routing in probabilistic graphs to minimize expected delay," in *Proc. Real-Time Syst. Symp. (RTSS)*, 2020, pp. 63–75.

[121] Z. Bo, Y. Qiao, C. Leng, H. Wang, C. Guo, and S. Zhang, "Developing real-time scheduling policy by deep reinforcement learning," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2021, pp. 131–142.

[122] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET benchmarks: Past, present and future," in *Proc. 10th Int. Workshop Worst-Case Execution Time Anal. (WCET)*, 2010, pp. 1–11.

[123] R. Wilhelm *et al.*, "The worst-case execution-time problem—Overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, p. 36, May 2008. [Online]. Available: https://doi.org/10.1145/1347375.1347389

[124] T. Huybrechts, S. Mercelis, and P. Hellinckx, "A new hybrid approach on WCET analysis for real-time systems using machine learning," in *Proc. 18th Int. Workshop Worst-Case Execution Time Anal. (WCET)*, 2018, pp. 1–12.

[125] T. Huybrechts, T. Cassimon, S. Mercelis, and P. Hellinckx, "Introduction of deep neural network in hybrid WCET analysis," in *Proc. Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, 2018, pp. 415–425.

[126] P. Altenbernd, J. Gustafsson, B. Lisper, and F. Stappert, "Early execution time-estimation through automatically generated timing models," *Real Time Syst.*, vol. 52, no. 6, pp. 731–760, Nov. 2016. [Online]. Available: https://doi.org/10.1007/s11241-016-9250-7

[127] A. Bonenfant, D. Claraz, M. de Michiel, and P. Sotin, "Early WCET prediction using machine learning," in *Proc. 17th Int. Workshop Worst-Case Execution Time Anal. (WCET)*, 2017, p. 5. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7307

[128] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey," *ACM Comput. Surveys*, vol. 52, no. 1, p. 14, 2020. [Online]. Available: https://doi.org/10.1145/3301283

[129] A. Burns, "Multi-model systems—An MCS by any other name," in *Proc. 8th Int. Workshop Mixed Criticality Syst.*, 2020, pp. 1–4.

[130] K. Agrawal, S. Baruah, and A. Burns, "Semi-clairvoyance in mixed-criticality scheduling," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2019, pp. 458–468.

[131] Z. Jiang, K. Yang, N. Fisher, N. Audsley, and Z. Dong, "Pythia-MCS: Enabling quarter-clairvoyance in I/O-driven mixed-criticality systems," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 38–50.

[132] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*.

[133] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[134] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[135] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" 2020, *arXiv:2003.03033*.

[136] Y. Guo, "A survey on methods and theories of quantized neural networks," 2018, *arXiv:1808.04752*.

[137] Q. Xing, M. Xu, T. Li, and Z. Guan, "Early exit or not: Resource-efficient blind quality enhancement for compressed images," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 275–292.

[138] J. Park *et al.*, "Faster CNNs with direct sparse convolutions and guided pruning," 2016, *arXiv:1608.01409*.

[139] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.

[140] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5687–5695.

[141] T. Vieira and J. Eisner, "Learning to prune: Exploring the frontier of fast and accurate parsing," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 263–278, Aug. 2017.

[142] F. Tung, S. Muralidharan, and G. Mori, "Fine-pruning: Joint fine-tuning and compression of a convolutional network with Bayesian optimization," 2017, *arXiv:1707.09102*.

[143] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015, *arXiv:1506.02626*.

[144] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5058–5066.

[145] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," 2016, *arXiv:1608.04493*.

[146] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1389–1397.

[147] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2015, *arXiv:1510.00149*.

[148] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 365–382.

[149] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard, "Adaptive quantization for deep neural network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–9.

[150] S. Ye *et al.*, "Progressive DNN compression: A key to achieve ultra-high weight pruning and quantization rates using admm," 2019, *arXiv:1903.09769*.

[151] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, p. 531.

[152] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," 2019, *arXiv:1908.09355*.

[153] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, "Face model compression by distilling knowledge from neurons," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 3560–3566.

[154] Z. Yang, L. Shou, M. Gong, W. Lin, and D. Jiang, "Model compression with two-stage multi-teacher knowledge distillation for Web question answering system," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 690–698.

[155] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4133–4141.

[156] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[157] A. A. Salah, E. Alpaydin, and L. Akarun, "A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 420–425, Mar. 2002.

[158] A. Wong, M. Famouri, and M. J. Shafiee, "AttendNets: Tiny deep image recognition neural networks for the edge via visual attention condensers," 2020, *arXiv:2009.14385*.

[159] Y.-F. Ma, X.-S. Hua, L. Lu, and H.-J. Zhang, "A generic framework of user attention model and its application in video summarization," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 907–919, Oct. 2005.

[160] A. M. Treisman, "Strategies and models of selective attention," *Psychol. Rev.*, vol. 76, no. 3, p. 282, 1969.

[161] A. H. van der Heijden, *Selective Attention in Vision*. London, U.K.: Routledge, 2003.

[162] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, "Sparse low rank factorization for deep neural network compression," *Neurocomputing*, vol. 398, pp. 185–196, 2020.

[163] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7370–7379.

[164] T. Chen, J. Lin, T. Lin, S. Han, C. Wang, and D. Zhou, "Adaptive mixture of low-rank factorizations for compact neural modeling," 2018, pp. 1–15.

[165] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic CNN compression via low-rank decomposition with knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 2889–2905, Dec. 2019.

[166] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[167] X. Ruan *et al.*, "EDP: An efficient decomposition and pruning scheme for convolutional neural network compression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4499–4513, Oct. 2021.

[168] A.-H. Phan *et al.*, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 522–539.

[169] M. Yin, Y. Sui, S. Liao, and B. Yuan, "Towards efficient tensor decomposition-based DNN model compression with optimization framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10674–10683.

[170] B. Wu, D. Wang, G. Zhao, L. Deng, and G. Li, "Hybrid tensor decomposition in neural network compression," *Neural Netw.*, vol. 132, pp. 309–320, Jun. 2020.

[171] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, P. Chuang, and L. Chang, "Compensated-DNN: Energy efficient low-precision deep neural networks by compensating quantization errors," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, 2018, pp. 1–6.

[172] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," 2017, *arXiv:1705.08665*.

[173] S. Lee and S. Nirjon, "Neuro.ZERO: A zero-energy neural network accelerator for embedded sensing and inference systems," in *Proc. 17th Conf. Embedded Netw. Sens. Syst.*, 2019, pp. 138–152.

[174] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[175] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4114–4122.

[176] K. Yang *et al.*, "cDeepArch: A compact deep neural network architecture for mobile sensing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 2043–2055, Oct. 2019.

[177] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2017, pp. 65–74.

[178] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[179] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2018, pp. 270–279.

[180] S. Kim, Y.-K. Noh, and F. C. Park, "Efficient neural network compression via transfer learning for machine vision inspection," *Neurocomputing*, vol. 413, pp. 294–304, Nov. 2020.

[181] M. A. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," 2020, *arXiv:2002.08307*.

[182] H.-C. Shin *et al.*, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[183] C. Zhong, X. Mu, X. He, J. Wang, and M. Zhu, "SAR target image classification based on transfer learning and model compression," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 3, pp. 412–416, Mar. 2019.

[184] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2016, pp. 475–480.

[185] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 2464–2469.

[186] K. Liao, Y. Zhang, X. Ren, Q. Su, X. Sun, and B. He, "A global past-future early exit method for accelerating inference of pre-trained language models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist. Human Lang. Technol.*, 2021, pp. 2013–2023.

[187] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," 2020, *arXiv:2006.04152*.

[188] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu, "A novel channel pruning method for deep neural network compression," 2018, *arXiv:1805.11394*.

[189] S. Bateni and C. Liu, "ApNet: Approximation-aware real-time neural network," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2018, pp. 67–79.

[190] S. Bateni, H. Zhou, Y. Zhu, and C. Liu, "Predjoule: A timing-predictable energy optimization framework for deep neural networks," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2018, pp. 107–118.

[191] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," 2017, *arXiv:1703.09844*.

[192] "The Difference Between Deep Learning Training and Inference: IntelAI." Intel. 2020. [Online]. Available: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/deep-learning-training-and-inference.html

[193] E. Lind and Ä. P. Velasquez, "A performance comparison between CPU and GPU in tensorflow," M.S. thesis, Skolan Elektroteknik Och Datavetenskap, KTH Royal Inst. Technol., Stockholm, Sweden, 2019.

[194] S. A. Mojumder *et al.*, "Profiling DNN workloads on a volta-based DGX-1 system," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, 2018, pp. 122–133.

[195] W. Li, G. Jin, X. Cui, and S. See, "An evaluation of unified memory technology on NVIDIA GPUs," in *Proc. 15th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2015, pp. 1092–1098.

[196] "NVIDIA Perfkit." NVIDIA Performance Toolkit. NVIDIA. 2014. [Online]. Available: https://developer.nvidia.com/nvidia-perfkit

[197] C. Li, A. Dakkak, J. Xiong, W. Wei, L. Xu, and W.-M. Hwu, "XSP: Across-stack profiling and analysis of machine learning models on GPUs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2020, pp. 326–327.

[198] S. Dey, J. Mondal, and A. Mukherjee, "Offloaded execution of deep learning inference at edge: Challenges and insights," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, 2019, pp. 855–861.

[199] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 31–36.

[200] P. Viola and M. Jones, "Robust real-time object detection," *Int. J. Comput. Vis.*, vol. 4, pp. 1–25, Jul. 2001.

[201] A. Grubb and D. Bagnell, "SpeedBoost: Anytime prediction with uniform near-optimality," in *Proc. Artif. Intell. Stat.*, 2012, pp. 458–466.

[202] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 572–579.

[203] K. Trapeznikov and V. Saligrama, "Supervised sequential classification under budget constraints," in *Proc. Artif. Intell. Stat.*, 2013, pp. 581–589.

[204] Z. Xu, K. Weinberger, and O. Chapelle, "The greedy miser: Learning under test-time budgets," 2012, *arXiv:1206.6451*.

[205] Z. Xu, M. Kusner, K. Weinberger, and M. Chen, "Cost-sensitive tree of classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 133–141.

[206] F. Nan, J. Wang, and V. Saligrama, "Feature-budgeted random forest," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1983–1991.

[207] J. Wang, K. Trapeznikov, and V. Saligrama, "Efficient learning by directed acyclic graph for resource constrained prediction," 2015, *arXiv:1510.07609*.

[208] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," 2016, *arXiv:1605.07648*.

[209] A. Graves, "Adaptive computation time for recurrent neural networks," 2016, *arXiv:1603.08983*.

[210] M. Figurnov *et al.*, "Spatially adaptive computation time for residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1039–1048.

[211] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for fast test-time prediction," 2017, *arXiv:1702.07811*.

[212] A. Odena, D. Lawson, and C. Olah, "Changing model behavior at test-time using reinforcement learning," 2017, *arXiv:1702.07780*.

[213] A. R. Zamir *et al.*, "Feedback networks," in *Proc. IEEE Conf. Computer Vis. Pattern Recognit.*, 2017, pp. 1308–1317.

[214] E. Bini, G. Buttazzo, and G. Lipari, "Minimizing CPU energy in real-time systems with discrete speed management," *ACM Trans. Embedded Comput. Syst.*, vol. 8, no. 4, pp. 1–23, 2009.

[215] A. Dudani, F. Mueller, and Y. Zhu, "Energy-conserving feedback EDF scheduling for embedded systems with real-time constraints," *ACM SIGPLAN Notices*, vol. 37, no. 7, pp. 213–222, 2002.

[216] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. Abdelzaher, and X. Liu, "OptiTuner: On performance composition and server farm energy minimization application," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1871–1878, Nov. 2011.

[217] H. Hoffmann, "CoAdapt: Predictable behavior for accuracy-aware applications running on power-aware systems," in *Proc. 26th Euromicro Conf. Real-Time Syst.*, 2014, pp. 223–232.

[218] M. Alcon, H. Tabani, L. Kosmidis, E. Mezzetti, J. Abella, and F. J. Cazorla, "Timing of autonomous driving software: Problem analysis and prospects for future solutions," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2020, pp. 267–280.

[219] W. Jang, H. Jeong, K. Kang, N. Dutt, and J.-C. Kim, "R-TOD: Real-time object detector with minimized end-to-end delay for autonomous driving," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 191–204.

[220] H. Zhang and H. Hoffmann, "Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 545–559, 2016.

[221] A. Farrell and H. Hoffmann, "MEANTIME: Achieving both minimal energy and timeliness with approximate computing," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2016, pp. 421–435.

[222] C. Imes, D. H. Kim, M. Maggio, and H. Hoffmann, "POET: A portable approach to minimizing energy under soft real-time constraints," in *Proc. 21st IEEE Real-Time Embedded Technol. Appl. Symp.*, 2015, pp. 75–86.

[223] N. D. Lane *et al.*, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sens. Netw. (IPSN)*, 2016, pp. 1–12.

[224] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "McDNN: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. 14th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2016, pp. 123–136.

[225] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.

[226] Y. Meidan *et al.*, "N-baIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul./Sep. 2018.

[227] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[228] B. Roy and H. Cheung, "A deep learning approach for intrusion detection in Internet of Things using bi-directional long short-term memory recurrent neural network," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, 2018, pp. 1–6.

[229] H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for Internet of Things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018.

[230] A. Dawoud, S. Shahristani, and C. Raun, "Deep learning and software-defined networks: Towards secure IoT architecture," *Internet Things*, vols. 3–4, pp. 82–89, Oct. 2018.

[231] Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2018, pp. 262–267.

[232] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," in *Proc. Int. ISCIS Security Workshop*, 2018, pp. 79–89.

[233] S. Homayoun *et al.*, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Gener. Comput. Syst.*, vol. 90, pp. 94–104, Jan. 2019.

[234] P. Koopman, "Embedded system security," *Computer*, vol. 37, no. 7, pp. 95–97, Jul. 2004.

[235] A. Ruhland, C. Prehofer, and O. Horst, "embSFI: An approach for software fault isolation in embedded systems," in *Proc. 1st Workshop Security Dependability Critical Embedded Real-Time Syst.*, 2016, pp. 6–11.

[236] M. Mamdouh, M. A. I. Elrukhsi, and A. Khattab, "Securing the Internet of Things and wireless sensor networks via machine learning: A survey," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, 2018, pp. 215–218.

[237] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 3rd Quart., 2020.

[238] H. Xiong, Y. Huang, L. E. Barnes, and M. S. Gerber, "Sensus: A cross-platform, general-purpose system for mobile crowdsensing in human-subject studies," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 415–426.

[239] Y. Huang *et al.*, "Assessing social anxiety using gps trajectories and point-of-interest data," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 898–903.

[240] H. Xiong, D. Zhang, L. Wang, J. P. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, p. 39, 2015.

[241] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4W1H in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 42–48, Aug. 2014.

[242] J. Bian, H. Xiong, Y. Fu, and S. Das, "CSWA: Aggregation-free spatial-temporal community sensing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, p. 254.

[243] J. Bian, H. Xiong, W. Cheng, W. Hu, Z. Guo, and Y. Fu, "Multi-party sparse discriminant learning," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2017, pp. 745–750.

[244] L. Wang *et al.*, "CCS-TA: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 683–694.

[245] J. Bian, H. Xiong, Y. Fu, J. Huan, and Z. Guo, "MP2SDA: Multi-party parallelized sparse discriminant learning," *ACM Trans. Knowl. Disc. Data*, vol. 14, no. 3, pp. 1–22, 2020.

[246] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "DeepPAR and DeepDPA: Privacy preserving and asynchronous deep learning for Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2081–2090, Mar. 2020.

[247] L. Li, H. Xiong, Z. Guo, J. Wang, and C.-Z. Xu, "Smartpc: Hierarchical pace control in real-time federated learning system," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2019, pp. 406–418.

[248] Z. Feng *et al.*, "SecureGBM: Secure multi-party gradient boosting," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2019, pp. 1312–1321.

[249] Q. Kang *et al.*, "Quasi-optimal data placement for secure multi-tenant data federation on the cloud," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2020, pp. 1954–1963.

[250] D. C. G. Valadares, A. A. de Carvalho Cesar Sobrinho, A. Perkusich, and K. C. Gorgonio, "Formal verification of a trusted execution environment-based architecture for IoT applications," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 17199–17210, Dec. 2021.

[251] J. Jang and B. B. Kang, "Securing a communication channel for the trusted execution environment," *Comput. Security*, vol. 83, pp. 79–92, Jun. 2019.

[252] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized IoT data management using blockchain and trusted execution environment," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, 2018, pp. 15–22.

[253] P. M. Shakeel, S. Baskar, V. R. S. Dhulipala, S. Mishra, and M. M. Jaber, "Maintaining security and privacy in health care system using learning based deep-Q-networks," *J. Med. Syst.*, vol. 42, no. 10, p. 186, 2018.

[254] B. Chatterjee, D. Das, S. Maity, and S. Sen, "RF-PUF: Enhancing IoT security through authentication of wireless nodes using in-situ machine learning," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 388–398, Feb. 2019.

[255] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2019, pp. 452–457.

[256] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial IoT using machine learning," in *Proc. IEEE Int. Conf. Intell. Security Inform. (ISI)*, 2018, pp. 112–117.

[257] W. Abbass, Z. Bakraouy, A. Baina, and M. Bellafkih, "Classifying IoT security risks using deep learning algorithms," in *Proc. 6th Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2018, pp. 1–6.

[258] Y. E. Sagduyu, Y. Shi, and T. Erpek, "IoT network security from the perspective of adversarial deep learning," in *Proc. 16th Annu. IEEE Int. Conf. Sens. Commun. Netw. (SECON)*, 2019, pp. 1–9.

[259] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?" *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 41–49, Sep. 2018.

[260] M. A. Amanullah *et al.*, "Deep learning and big data technologies for IoT security," *Comput. Commun.*, vol. 151, pp. 495–517, Feb. 2020.

[261] S. M. Tahsien, H. Karimipour, and P. Spachos, "Machine learning based solutions for security of Internet of Things (IoT): A survey," *J. Netw. Comput. Appl.*, vol. 161, Jul. 2020, Art. no. 102630.

[262] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100227.

[263] K. D. Ahmed and S. Askar, "Deep learning models for cyber security in IoT networks: A review," *Int. J. Sci. Bus.*, vol. 5, no. 3, pp. 61–70, 2021.

[264] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, "A systematic review on deep learning approaches for IoT security," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100389.

[265] C. Gongye, Y. Fei, and T. Wahl, "Reverse-engineering deep neural networks using floating-point timing side-channels," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–6.

[266] X. Hu *et al.*, "DeepSniffer: A DNN model extraction framework based on learning architectural hints," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2020, pp. 385–399. [Online]. Available: https://doi.org/10.1145/3373376.3378460

[267] W. Y. B. Lim *et al.*, "When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 457–466, Jan. 2022.

[268] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[269] J. Kang *et al.*, "Optimizing task assignment for reliable blockchain-empowered federated edge learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1910–1923, Feb. 2021.

[270] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6G communications: Challenges, methods, and future directions," *China Commun.*, vol. 17, no. 9, pp. 105–118, Sep. 2020.

[271] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.

[272] Y. Qu, C. Dong, J. Zheng, H. Dai, F. Wu, S. Guo, and A. Anpalagan, "Empowering edge intelligence by air-ground integrated federated learning," *IEEE Netw.*, vol. 35, no. 5, pp. 34–41, Sep./Oct. 2021.

[273] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–3, 1st Quart., 2022.

[274] J. Nagi, K. S. Yap, S. K. Tiong, S. K. Ahmed, and A. Mohammad, "Detection of abnormalities and electricity theft using genetic support vector machines," in *Proc. IEEE Region 10 Conf.*, 2008, pp. 1–6.

[275] M. Mishra, J. Nayak, B. Naik, and A. Abraham, "Deep learning in electrical utility industry: A comprehensive review of a decade of research," *Eng. Appl. Artif. Intell.*, vol. 96, Nov. 2020, Art. no. 104000.

[276] T. Han, K. Muhammad, T. Hussain, J. Lloret, and S. W. Baik, "An efficient deep learning framework for intelligent energy management in IoT networks," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3170–3179, Mar. 2021.

[277] N. Dey, S. Fong, W. Song, and K. Cho, "Forecasting energy consumption from smart home sensor network by deep learning," in *Proc. Int. Conf. Smart Trends Inf. Technol. Comput. Commun.*, 2017, pp. 255–265.

[278] S. A. Boyer, *SCADA: Supervisory Control and Data Acquisition*. Durham, NC, USA: Int. Soc. Autom., 2009.

[279] K. T. Erickson, "Programmable logic controllers," *IEEE Potentials*, vol. 15, no. 1, pp. 14–17, Feb./Mar. 1996.

[280] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Qual. Rel. Eng. Int.*, vol. 17, no. 2, pp. 105–112, 2001.

[281] J. Liu, D. Djurdjanovic, K. A. Marko, and J. Ni, "A divide and conquer approach to anomaly detection, localization and diagnosis," *Mech. Syst. Signal Process.*, vol. 23, no. 8, pp. 2488–2499, 2009.

[282] Y. Jiang, W. Wang, and C. Zhao, "A machine vision-based realtime anomaly detection method for industrial products using deep learning," in *Proc. Chin. Autom. Congr. (CAC)*, 2019, pp. 4842–4847.

[283] C. M. Ryan, A. Parnell, and C. Mahoney, "Real-time anomaly detection for advanced manufacturing: Improving on Twitter's state of the art," 2019, *arXiv:1911.05376*.

[284] P. Ferrari *et al.*, "Performance evaluation of full-cloud and edge-cloud architectures for Industrial IoT anomaly detection based on deep learning," in *Proc. II Workshop Metrology Ind. 4.0 IoT (MetroInd4. 0 IoT)*, 2019, pp. 420–425.

[285] I. Lawrence and K. Lin, "Assay validation using the concordance correlation coefficient," *Biometrics*, vol. 48, no. 2, pp. 599–604, 1992.

[286] H.-H. Hsu and C.-C. Chen, "Rfid-based human behavior modeling and anomaly detection for elderly care," *Mobile Inf. Syst.*, vol. 6, no. 4, pp. 341–354, 2010.

[287] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Rule-based anomaly pattern detection for detecting disease outbreaks," in *Proc. AAAI/IAAI*, 2002, pp. 217–223.

[288] H. Ghayvat, S. Pandya, and A. Patel, "Deep learning model for acoustics signal based preventive healthcare monitoring and activity of daily living," in *Proc. 2nd Int. Conf. Data Eng. Appl. (IDEA)*, 2020, pp. 1–7.

[289] F. Ali *et al.*, "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Inf. Fusion*, vol. 63, pp. 208–222, Nov. 2020.

[290] X. Wu, C. Liu, L. Wang, and M. Bilal, "Internet of Things-enabled real-time health monitoring system using deep learning," *Neural Comput. Appl.*, pp. 1–12, Sep. 2021.

[291] M. Parsa, P. Panda, S. Sen, and K. Roy, "Staged inference using conditional deep learning for energy efficient real-time smart diagnosis," in *Proc. 39th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, 2017, pp. 78–81.

[292] S. Tennyson and P. Salsas-Forn, "Claims auditing in automobile insurance: Fraud detection and deterrence objectives," *J. Risk Insurance*, vol. 69, no. 3, pp. 289–308, 2002.

[293] S. Viaene, R. A. Derrig, B. Baesens, and G. Dedene, "A comparison of state-of-the-art classification techniques for expert automobile insurance claim fraud detection," *J. Risk Insurance*, vol. 69, no. 3, pp. 373–421, 2002.

[294] M. Artís, M. Ayuso, and M. Guillén, "Detection of automobile insurance fraud with discrete choice models and misclassified claims," *J. Risk Insurance*, vol. 69, no. 3, pp. 325–340, 2002.

[295] Y. Wang and W. Xu, "Leveraging deep learning with lda-based text analytics to detect automobile insurance fraud," *Decis. Support Syst.*, vol. 105, pp. 87–95, Jan. 2018.

[296] R. Singh, M. P. Ayyar, T. V. S. Pavan, S. Gosain, and R. R. Shah, "Automating car insurance claims using deep learning techniques," in *Proc. IEEE 5th Int. Conf. Multimedia Big Data (BigMM)*, 2019, pp. 199–207.

[297] J. J.-C. Ying, P.-Y. Huang, C.-K. Chang, and D.-L. Yang, "A preliminary study on deep learning for predicting social insurance payment behavior," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2017, pp. 1866–1875.

[298] E. Priyanka *et al.*, "Real-time performance analysis of multiple parameters of automotive sensor's can data to predict vehicle driving efficiency," *Int. J. Comput. Digit. Syst.*, vol. 11, no. 1, pp. 1–21, 2021.

[299] S. P. Sethi, H. Yan, and H. Zhang, *Inventory and Supply Chain Management With Forecast Updates*, vol. 81. Cham, Switzerland: Springer, 2005.

[300] Y. M. Lee, F. Cheng, and Y. T. Leung, "Exploring the impact of RFID on supply chain dynamics," in *Proc. Winter Simulat. Conf.*, vol. 2, 2004, pp. 1145–1152.

[301] B. Sahay and J. Ranjan, "Real time business intelligence in supply chain analytics," *Inf. Manage. Comput. Security*, vol. 16, no. 1, pp. 28–48, 2008.

[302] E. Peters, T. Kliestik, H. Musa, and P. Durana, "Product decision-making information systems, real-time big data analytics, and deep learning-enabled smart process planning in sustainable industry 4.0," *J. Self-Governance Manage. Econ.*, vol. 8, no. 3, pp. 16–22, 2020.

[303] M. Cherrington, Z. J. Lu, Q. Xu, D. Airehrour, S. Madanian, and A. Dyrkacz, "Deep learning decision support for sustainable asset management," in *Advances in Asset Management and Condition Monitoring*. Cham, Switzerland: Springer, 2020, pp. 537–547.

[304] R. Gonzalez *et al.*, "Net2Vec: Deep learning for the network," in *Proc. Workshop Big Data Anal. Mach. Learn. Data Commun. Netw.*, 2017, pp. 13–18.

[305] N. Kargah-Ostadi, A. Waqar, and A. Hanif, "Automated real-time roadway asset inventory using artificial intelligence," *Transp. Res. Rec.*, vol. 2674, no. 11, pp. 220–234, 2020.

[306] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, "Real-time system for monitoring driver vigilance," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 63–77, Mar. 2006.

[307] Z. Zhu and Q. Ji, "Real time and non-intrusive driver fatigue monitoring," in *Proc. 7th Int. IEEE Conf. Intell. Transp. Syst.*, 2004, pp. 657–662.

[308] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real Time Imag.*, vol. 8, no. 5, pp. 357–377, 2002.

[309] M. J. Flores, J. M. Armingol, and A. de la Escalera, "Real-time warning system for driver drowsiness detection using visual information," *J. Intell. Robot. Syst.*, vol. 59, no. 2, pp. 103–125, 2010.

[310] D. Tran, H. M. Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," *IET Intell. Transp. Syst.*, vol. 12, no. 10, pp. 1210–1219, 2018.

[311] A. Theofilatos, C. Chen, and C. Antoniou, "Comparing machine learning and deep learning methods for real-time crash prediction," *Transp. Res. Rec.*, vol. 2673, no. 8, pp. 169–178, 2019.

[312] X. Wang, W. Zhang, X. Wu, L. Xiao, Y. Qian, and Z. Fang, "Real-time vehicle type classification with deep convolutional neural networks," *J. Real-Time Image Process.*, vol. 16, no. 1, pp. 5–14, 2019.

[313] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time attack-recovery for cyber-physical systems using linear approximations," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 205–217.

[314] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[315] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[316] N. Otterness, V. Miller, M. Yang, J. Anderson, F. D. Smith, and S. Wang, "GPU sharing for image processing in embedded real-time systems," in *Proc. OSPERT*, 2016, pp. 1–7.

[317] H. Zhou, S. Bateni, and C. Liu, "S^ 3DNN: Supervised streaming and scheduling for GPU-accelerated real-time DNN workloads," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2018, pp. 190–201.

[318] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "GPU scheduling on the NVIDIA TX2: Hidden details revealed," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2017, pp. 104–115.

[319] Y. Xiang and H. Kim, "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2019, pp. 392–405.

[320] Y. Bao, Y. Peng, Y. Chen, and C. Wu, "Preemptive all-reduce scheduling for expediting distributed DNN training," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 626–635.

[321] Y. Choi and M. Rhu, "PREMA: A predictive multi-task scheduling algorithm for preemptible neural processing units," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2020, pp. 220–233.

[322] O. Wechsler, M. Behar, and B. Daga, "Spring hill (NNP-I 1000) Intel's data center inference chip," in *Proc. IEEE Hot Chips 31st Symp. (HCS)*, 2019, pp. 1–12.

[323] J.-H. Jacobsen, E. Oyallon, S. Mallat, and A. W. Smeulders, "Multiscale hierarchical convolutional networks," 2017, *arXiv:1703.04140*.

**Jiang Bian** (Member, IEEE) received the B.Eng. degree in logistics systems engineering from Huazhong University of Science and Technology, Wuhan, China, in 2014, the M.Sc. degree in industrial systems engineering from the University of Florida, Gainesville, FL, USA, in 2016, and the Ph.D. degree in computer engineering from the University of Central Florida, Orlando, FL, USA, in 2020.

He is currently with the Big Data Laboratory, Baidu Research, Beijing, China. His research interests include ubiquitous computing and AutoDL.

**Abdullah Al Arafat** (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2016, and the M.S. degree in computer engineering from the University of Central Florida, Orlando, FL, USA, in 2020, where he is currently pursuing the Ph.D. degree in computer engineering.

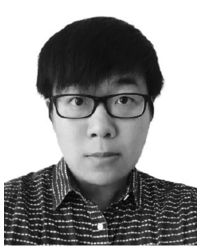
His research interests include scheduling theory, algorithms, and real-time and intelligent systems.

**Haoyi Xiong** (Senior Member, IEEE) received the Ph.D. degree in computer science from Telecom SudParis jointly from Université Pierre et Marie Curie, Évry, France, in 2015.

From 2016 to 2018, he was a Tenure-Track Assistant Professor with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA, and a Postdoctoral Fellow with the University of Virginia, Charlottesville, VA, USA, from 2015 to 2016. He is currently a Principal Research Scientist with the Big Data Laboratory, Baidu Research, Beijing, China, and also a Graduate Faculty Scholar affiliated with the University of Central Florida, Orlando, FL, USA. He has published more than 70 papers in top computer science conferences and journals, including UbiComp, ICML, ICLR, KDD, RTSS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and IEEE INTERNET OF THINGS JOURNAL. His current research interests include AutoDL and ubiquitous computing.

**Jing Li** (Member, IEEE) received the B.S. degree in computer science from Harbin Institute of Technology, Harbin, China, in 2011, and the Ph.D. degree from Washington University in St. Louis, St. Louis, MO, USA, in 2017, under the guidance of Prof. C. Lu and K. Agrawal.

She is an Assistant Professor with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. She has high-impact publications in top journals and conferences with three outstanding paper awards. Her research interests include real-time systems, parallel computing, cyber–physical systems, and reinforcement learning for system design and optimization.



**Li Li** (Member, IEEE) received the B.S. degree from Tianjin University, Tianjin, China, in 2011, and the M.S. and Ph.D. degrees in electrical and computer engineering from Ohio State University, Columbus, OH, USA, in 2014 and 2018, respectively.

He is currently an Assistant Professor with the University of Macau, Macau, China. He has published in refereed journals and conference proceedings, such as INFOCOM, RTSS, ICDCS, NDSS, MM, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.

**Hongyang Chen** (Senior Member, IEEE) received the B.S. and M.S. degrees from Southwest Jiaotong University, Chengdu, China, in 2003 and 2006, respectively, and the Ph.D. degree from The University of Tokyo, Tokyo, Japan, in 2011.

From 2011 to 2020, he was a Researcher with Fujitsu Ltd., Tokyo. He is an Adjunct Professor with Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou, China. He is currently a Senior Research Expert with Zhejiang Laboratory, Hangzhou. He has authored or coauthored 100+ refereed journals and conference papers in top venues, and has been granted or filed more than 50 PCT patents. His research interests include the IoT, data-driven intelligent networking and systems, machine learning, localization, location-based big data, B5G, and statistical signal processing.

**Jun Wang** (Fellow, IEEE) received the B.Eng. degree in computer engineering from Wuhan Technical University of Surveying and Mapping (currently, Wuhan University), Wuhan, China, the M.Eng. degree in computer Engineering from Huazhong University of Science and Technology, Wuhan, and the Ph.D. degree in computer Science and engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2002.

He is a Professor of Computer Engineering and the Director of the Computer Architecture and Storage Systems Laboratory, University of Central Florida, Orlando, FL, USA. He has conducted extensive research in the areas of computer systems and data-intensive computing. He has authored over 150 publications in premier journals and conferences. His specific research interests include massive storage and file systems in a local, distributed, and parallel systems environment.

Dr. Wang is the recipient of the National Science Foundation Early Career Award 2009 and the Department of Energy Early Career Principal Investigator Award 2005.

**Dejing Dou** (Senior Member, IEEE) received the bachelor's degree from Tsinghua University, Beijing, China, in 1996, and the Ph.D. degree from Yale University, New Haven, CT, USA, in 2004.

He is the Head of the Big Data Laboratory and Business Intelligence Laboratory, Baidu Research, Beijing, China. He is also a Full Professor (on leave) with the Computer and Information Science Department, University of Oregon, Eugene, OR, USA, and has been leading Advanced Integration and Mining Laboratory since 2005. He has been the Director of the NSF IUCRC Center for Big Learning, Gainesville, FL, USA, since 2018. He was a Visiting Associate Professor with the Stanford Center for Biomedical Informatics Research, Stanford, CA, USA, from 2012 to 2013. He has published more than 100 research papers. His research areas include artificial intelligence, data mining, data integration, NLP, and health informatics.

**Zhishan Guo** (Senior Member, IEEE) received the B.Eng. degree (with Hons.) in computer science and technology from Tsinghua University, Beijing, China, in 2009, the M.Phil. degree in mechanical and automation engineering from The Chinese University of Hong Kong, Hong Kong, in 2011, and the Ph.D. degree in computer science from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, in 2016.

He is an Assistant Professor with the Department of Computer and Electrical Engineering, University of Central Florida, Orlando, FL, USA. His current research interests include real-time and cyber–physical systems, neural networks, and computational intelligence.

Dr. Guo has received Best Paper Awards from flagship conferences, such as RTSS and EMSOFT.