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

Jiang Bian†, Abdullah Al Arafat†, Haoyi Xiong*, Jing Li, Li Li,
Hongyang Chen, Jun Wang, Dejing Dou, and Zhishan Guo*

***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 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 community address the strict timing requirements of ML/DL technologies to include in real-time systems. This paper will rigorously explore the state-of-the-art results emphasizing the strengths and weaknesses in ML/DL-based scheduling techniques, accuracy vs. execution time trade-off policies of ML algorithms, and security & privacy of learning-based algorithms in real-time IoT systems.**

***Index Terms*—Internet of Things, Machine learning, Deep learning, Scheduling, Real-time systems.**

I. Introduction

Real-time systems (RTS) design must have both functional and temporal correctness [1], [2]. Thus, real-time systems 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: a) it requires a heterogeneous hardware platform consisting of CPUs (multi-core), GPUs, or specialized accelerators [4], which complicate the resource-sharing models of RTS. b) 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 (e.g., deep neural network (DNN)) have extensively progressed and are employed in enormous applications unprecedentedly over the last decade. So, machine learning 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

This research was supported, in part, by the National Science Foundation (USA) under Grant Numbers CNS–1948457, CNS–1850851, PPoSS-2028481, and OIA–1937833.

Abdullah Al Arafat, Jun Wang and Zhishan Guo (and Jiang Bian) are (were) with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA (email: {abdullah.arafat, bjbj11111}@Knights.ucf.edu, {Jun.Wang,zsguo}@ucf.edu).

Jiang Bian, Haoyi Xiong and Dejing Dou are with Big Data Lab, Baidu Inc., Beijing, China (email: {bianjiang03,xionghaoyi,doudejing}@baidu.com).

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

Hongyang Chen is with Research Center for Intelligent Network, Zhejiang Lab, Hangzhou, Zhejiang, China. (email: dr.h.chen@ieee.org).

Li Li is with Department of Computer and Information Science, University of Macau, Tapia, Macao (email: li.li@siat.ac.cn).

† The first two authors contributed equally to this work.

* Corresponding Authors: Zhishan Guo and Haoyi Xiong.

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:

- 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.
- ML algorithms for schedulability analysis — We review the papers that address the 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.
- Privacy and security in real-time system — We further review the works related to the privacy and security of RTS through deploying ML algorithms.
- Finally, we present several open problems and potential applications of ML in the real-time system that are yet to explore.

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

Organization. The rest of the paper is organized as follows. Section II discusses the background of real-time systems, specifically the concepts related to the scheduling of real-time systems and the learning algorithms commonly/potentially employed in real-time systems. Section III comprehensively surveys the existing works on machine learning in real-time IoTs. Section IV presents the applicability of machine learning algorithms in real-time IoT systems. Section V points out research gaps and proposes possible employment of machine learning in real-time IoTs. Finally, the paper 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 machine learning/deep learning 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 real-time system 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 and worst-case execution-time (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 — offline verification/certification and 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., pseudo-polynomial) time complexity of the schedulability test. The scheduling algorithms are designed for two different types of priority — static priority (task-level fixed priority), where the priority of each task in the task set is fixed; and 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 non-preemptive algorithms, such as Earliest Deadline First (EDF) [5], Rate-Monotonic (RM) [2], and 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 over-executes 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

Machine learning includes a wide range of algorithms from end-to-end problem-solving algorithms to specific feature extraction algorithms [11]–[15]. Machine learning algorithms are typically categorized into three directions: supervised learning [16], [17], unsupervised learning [18], [19], and 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 machine learning 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 hyper-parameter tuning on real-time embedded systems [22], real-time traffic classification through semi-supervised learning [23], and real-time inference and training for deep learning [24]. Apart from the classical categorization of machine learning algorithms, we purposely divide the machine learning algorithms applied in real-time IoT systems into two branches, which are statistical learning algorithms and neural network-based learning/deep learning algorithms. The motivation behind is to comprehensively review the previous efforts of applying learning algorithms in real-time systems 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 machine learning. 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 machine learning 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 machine learning 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 subsections, 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 machine learning 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 machine learning 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 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 machine learning 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

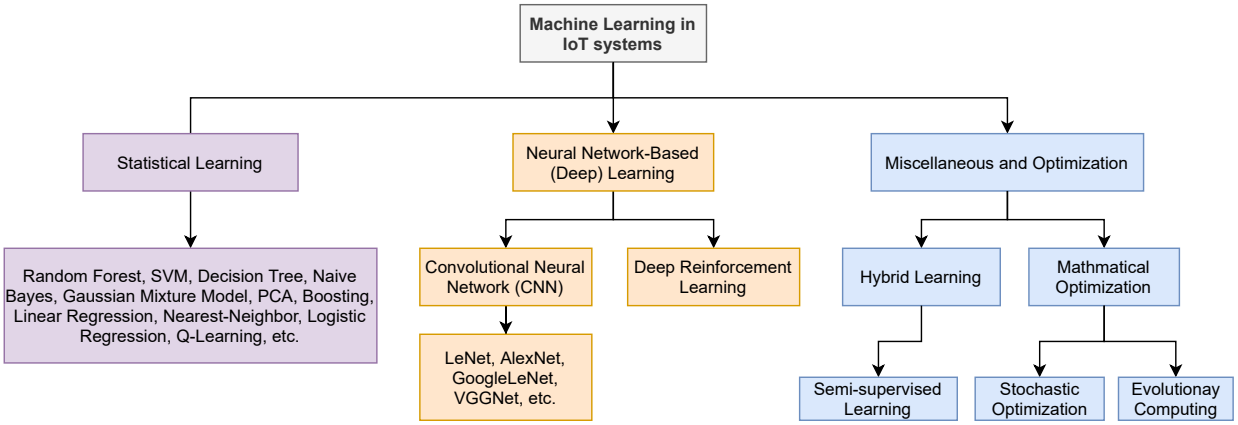


Fig. 1: Some Popular Machine Learning Branches in Real-Time IoT Systems.

will systematically review and summarize the employment of statistical machine learning algorithms in real-time IoT systems in Section III.

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

a) Convolutional Neural Networks: Convolutional neural network (CNN) is a class of deep learning algorithms, highly used in high-dimensional datasets such as images to extract low-dimensional latent space representations and location invariant features [77]. CNN is the building block of the famous deep learning architectures such as LeNet [78], AlexNet [79], GoogLeNet [80], VGGNet [81], etc. CNN is consists of several types of layers, such as Convolution Layers, Pooling Layers, Activation Layers, Fully-connected layers, etc. As one of the most common-used modules in deep learning/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.B.1.

b) Deep Reinforcement Learning: Reinforcement learning is a branch of machine learning 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 pre-defined final state, as illustrated in Figure 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.

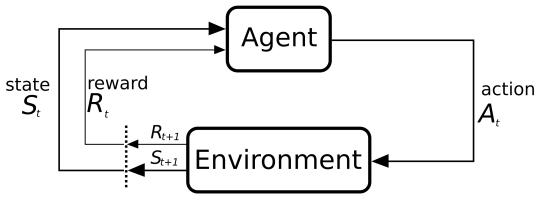


Fig. 2: Reinforcement learning framework.

However, due to its limitation on large-scale dynamic data-environment [21], [82], reinforcement learning intends to embrace the hug of deep learning in nowadays real-time IoT applications. 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 nets 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 Figure 3. That is, the UVA leverage the target image caption as the reward for movement guiding and wildlife tracking.

Rather than the above tasks in machine learning, deep reinforcement learning is also a powerful tool to solve combinatorial optimization and scheduling problems. These problems

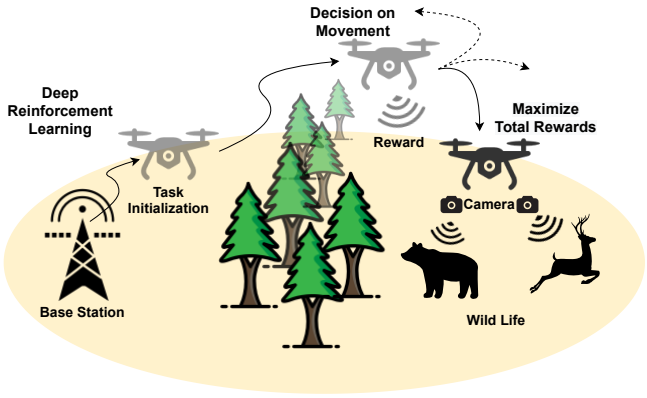


Fig. 3: Deep Convolutional Agent on UAV.

are often challenging in real-time systems and IoT applications, e.g., learning near-optimal schedules when the real-time system 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 machine learning 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 machine learning algorithms, we can also observe the rising of hybrid learning strategies and optimization algorithms in modern real-time IoT systems. For hybrid learning strategies, semi-supervised learning [88], [89] is one of the popular directions. In semi-supervised 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 semi-supervised 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 real-time system settings. For example, when handling the real-time streaming data, semi-supervised 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 (father and farther from the true manifold of data itself).

Another branch is the well-known optimization strategies (aka mathematical optimization) [92]–[94] which are not typically categorized into traditional machine learning algorithms. Although we often rely on them to solve the machine learning problems, e.g., stochastic optimization [95], [96] for Back-propagation in training neural networks, the optimization algorithms could be independent of machine learning 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 machine learning, deep learning 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 hard-ware 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 **i**) ML-based learning algorithms for real-time scheduling, **ii**) adaptation of ML-based learning algorithm to make it schedulable and **iii**) rising security issues when involving ML-based learning algorithms in real-time IoT systems. As illustrated in Figure 4, a horizontal tree-like structure shows the 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 subsection will explore the employment and applications, which consists of the most popular and exploited leaf topics as follows,

- ML for Scheduling Analysis in Real-Time IoT Systems.
  - ML-based Schedulability Analysis.
  - ML-based WCET Estimations.
  - ML-based System-behavior Prediction.
- Adaptation of ML in Real-Time IoT Systems.
  - Model Compression for Real-Time Performance.
  - Real-Time Pipeline.
- Security of Real-Time IoT Systems.

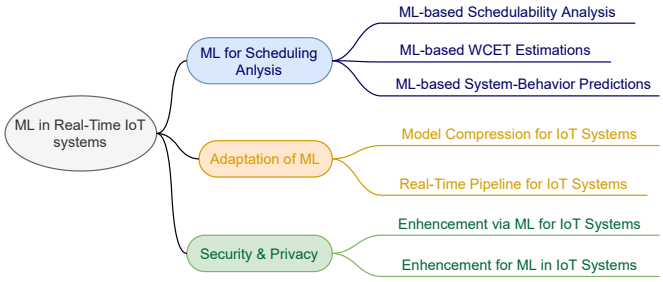


Fig. 4: The tree structure of ML employment.

For each topic, we will discuss the motivation, scope, technical details, contribution and remaining challenges of the related machine learning 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 multi-processor 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, machine learning 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, such as —

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

In [114], Lee et al. 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 real-time system 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 real-time system 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 subsections.

1) ML-Based Schedulability Analysis: In real-time systems, 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 worst-case execution time 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 IoTs, 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. In [118], Dziurzanski et al. 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 real-time systems' 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.

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] formulate 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 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. developed regression algorithms [124] and deep learning 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 system architecture. Therefore, these methods are not applicable in safety-critical hard-real-time systems.

In [9], Vestal presents a varying WCET-based scheduling technique called mixed-criticality systems to avoid very pessimistic WCET's of tasks in complex systems. As the WCET estimation became difficult, the mixed-criticality systems used different levels of WCET's 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 real-time systems are designed with deterministic behavior to guarantee the task meets every deadline. The safety-critical systems are traditionally designed as mixed-criticality systems [9] or multi-model systems [129]. In mixed-criticality systems, 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 (non-clairvoyant). 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 non-clairvoyant 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 non-clairvoyant systems. However, Pythia-MCS did not use any ML in their design, it may be possible to use 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 deep learning to embrace the explosive growth of data scale and problem complexity in 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 real-time systems, 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 i) compressing the model for implementation speed-up and ii) 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 subsections.

1) Model Compression for Real-Time IoT Systems.: DNNs with deep learning brings a revolution in the broad domain of computer vision and natural language processing (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 are 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 deep learning 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 over parametrized 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 $30\times$ to $50\times$ more compact AlexNet and VGG-16 models that have $3\times$ to $4\times$ layerwise speedup and $3\times$ to $7\times$ 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], and 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:

- **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 Figure 5, the number of synapses and neurons are in some degree reduced in the pruning process.

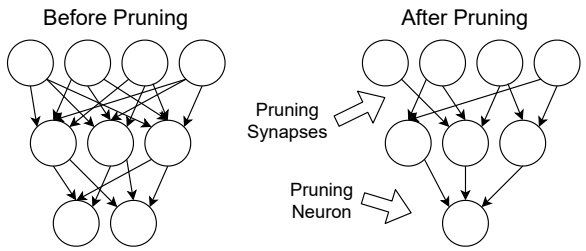


Fig. 5: Pruning on Synapses and Neuron.

- **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 are decreased, the size of the deep neural network can be significantly shrunken.
- **Knowledge Distillation.** Knowledge distillation [152]–[155] is often used in model training with large-scale dataset. 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.
- **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.
- **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

Techniques	Pros	Cons
Pruning	• Applicable in the process of training and post-training • Applicable to fully connected networks or convolutional networks (layers) • Balance the trade-off between inference time (model size) with accuracy [135]	• It is not as helpful as replacing with a better architecture [135] • 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 • 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] • 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) • 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 • Smaller model (e.g. a face detector and cropper could be only 44 KB) • 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 • Applicable to fully connected networks or convolutional networks (layers) • When applied in the process of training, it can reduce training time	• Computationally expensive • Cannot perform global parameters compression • 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 • It is flexible to use any number of bits to perform the DNN operations • 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 • 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 • No more training from scratch saves a lot of time and computation resource	• The target domain needs to be similar with the initial domain • Can not remove layers with confidence to reduce the number of parameters
Early Exit	• Applicable during and after training • Availability of multiple exit points depending on the goal of tasks • 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]

TABLE I: Main Stream Model Compression Techniques with their Pros and Cons.

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 convolutional neural networks.

- **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 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.
- **Transfer Learning.** Transfer Learning [178], [179] is another branch of machine learning 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 deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems, where the pre-trained 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]
- **Early Exit.** While DNNs benefit from a vast number of layers, it's 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] observe 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. Surat et al. [185] investigate 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 pre-trained model, where VGG16 model trained on the ImageNet dataset 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 pre-trained 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., deep learning 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 multi-process 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,

- Profiling ML models
- Selecting proper ML model for each task
- 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 feedback 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 feed-backs through a so-called back-propagation, where the weights are adjusted to 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's 20 billion 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.B.2.

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.

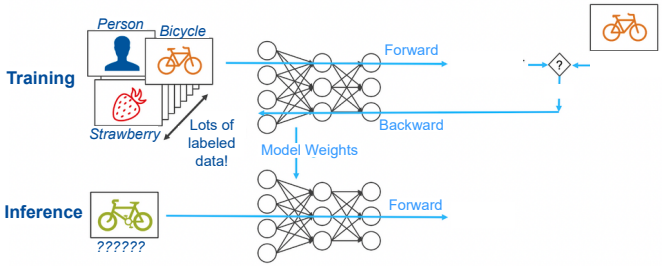


Fig. 6: The Training and Inference process of DNN [192]

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 multi-task 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 pre-processing, 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. propose an across-stack profiling design named XSP [197] 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 sub-optimal 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: I) 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; II) 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 machine learning 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 sub-modules 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 deep

learning inference tasks (some take care of both training and inference phase), the system-level objectives are far from being accomplished since a trade-off 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 multi-core 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], [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: a) on-demand resource allocation for capture, b) zero-slack pipeline, and c) 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. [189], [190] demonstrate 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], [190] propose a timing-predictable runtime system that guarantees hard deadlines of DNN workloads via efficient approximation. They design a layer-aware sub-deadline 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. [189] design and implement a runtime system for enhanced runtime multitasking performance, which exploits the mutually supplementary relation between approximation and resource sharing.

Although [189], [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 current system status). Thus, such a scenario leaves a future direction to design a clairvoyance system that can not only statically pre-schedule (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, 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 roles 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, the authors of [226] have proposed the use of deep autoencoders (DAE) to detect anomalous network traffic from compromised IoT devices. Deep learning 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 deep learning approach based on Stochastic Gradient Descent (SGD), which enables the detection of attacks in the social IoT. Besides, the authors of study [228] have proposed a deep learning technique that enables intrusion detection in IoT networks using the Bidirectional LSTM Recurrent Neural Network (BLSTM RNN). Furthermore, in study [229] the authors have proposed a deep learning 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. The authors of study [230] have introduced a framework for IoT based on Software Defined Networking (SDN). In study [231] the authors have discussed that IoT applications face major security issues in confidentiality, integrity, privacy, and availability. The

authors of study [232] have proposed a deep learning approach with Dense Random Neural networks (DRNN) 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 inter-connected with inter-operations, privacy and security have become critical issues of real-time IoT systems with machine learning and data analytic components [236], [237]. Generally, we categorize the existing works into two folders as follows.

- **Privacy and Security Enhancements for Data Processing and Machine Learning over IoTs.** As IoT devices, such as healthcare IoTs 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 machine learning or data processing algorithms on these devices. As early as [240], privacy preservation in mobile crowdsourced sensor networks has been studied, where 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 IoTs, aggregation-free distributed sensing has been proposed in [242] with mobile sensors, where authors proposed using decentralized SGD with multi-party computation [243] to enhance the compressive crowdsensing algorithms [244] for spatial-temporal monitoring. Besides data aggregation, machine learning for statistical supervised learning has been improved and secured for distributed IoT applications in [243], [245]. Zhang et al. [246] proposed DeepPar– a privacy preserving and asynchronous deep learning for industrial IoT over distributed datasets. For similar purposes, Li et al. [247] proposed SmartPC that secures the privacy of distributed datasets in a federated learning framework while minimizing the overall energy consumption on nodes. Rather than the separation of samples over distributed datasets, 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 (TEE) [249]–[252] is yet another way to perform data aggregation and machine learning using trustworthy infrastructures.
- **Privacy and Security Enhancements for IoT Systems using Data Analytics and Machine Learning.** In addition to securing the privacy and security of machine learning tasks over IoT systems, machine learning techniques could also enhance the privacy and security of real-time IoTs. Specifically, [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 machine learning algorithms. Roopak et al. [255] reviewed deep neural networks used in IoT cyber security. Zolanvari et al. [256] studied the effects of imbalanced datasets on IoT security with machine learning. The work [257] studied machine learning 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 IoTs. More work could be found in following surveys and technical reviews [259]–[264].

In addition to the above privacy and security issues for applications of Real-time IoTs with Machine Learning, some recent works have demonstrated the possibility to attack real-time schedulers for IoTs through reversing the execution times and orders of tasks [102], [265], [266] using machine learning 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 reliable and sustainable IoT system, a series of works study federated learning with respect to different privacy protection mechanisms. [267], [268] intend 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-arts 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 machine learning techniques in modern real-time IoT systems in terms of industrial practice.

Firstly, 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.

- 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.

- 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 1) the prediction accuracy increases by ML solutions, and 2) the automation is drastically improved by ML solutions in the manufacturing management.
- 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.
- 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.
- 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 pre-allocate 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.
- 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, subways timing). With 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. Towards Machine Learning for Real-Time Systems

As real-time systems have started to be used in various applications, real-time IoT design faces unprecedented challenges. 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 machine learning algorithms.

- **Predictability.** Predictability is the expected behavior of real-time systems. 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 real-time systems. The Mixed-criticality system design community has already explored probabilistic system behaviors for system mode-switch prediction. Predictability analysis can generally be performed by leveraging machine learning algorithms on cache/memory or I/O data access patterns and throughput analysis.
- **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 runtime 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. Machine learning-based algorithms can play a vital role in fast malicious attack detection methods with tolerable false-positive results.
- **Real-Time System Recovery.** Unlike general-purpose computing systems, real-time system 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.

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

Industries	Problems	Facilities	Solutions	Applicability
Utilities (energy, water, gas, and etc.)	• Real-time collection of usage data • Demand-supply prediction • Load balancing • 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]) • 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 • Production line automation	• Supervisory control and data acquisition systems (SCADA) [278] • Programmable logic controllers (PLCs) [279] • Cameras • IoT devices mounted or embedded	• Traditional: Anomaly detection via statistical strategies (e.g., chi-squrare [280], divide and conquer [281]). • 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 • Chronic disease management • Elderly care • Wellness and fitness programs	• Wearable and personal medical devices • Smart mobile phones	• Traditional: Historical correlation analysis via statistical tools [285]–[287]. • 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) • Prediction of property damage or rate of depreciation • 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]. • 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) • 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] • 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 • Asset management and tracking	• On-board vehicle gateway devices • RFID tags • 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] • ML/DL-based: Visualization, prediction, optimization, and decision support systems for associated transportation systems via ML/DL [310]–[312]	• Applicable in real-time manner.

TABLE II: Applicability of Machine Learning Techniques in Real-Time IoT Systems/Applications.

Machine learning can be used with more research on this perspective.

B. Towards Real-Time and Schedulable Machine learning

On the contrary, the deployment of machine learning or deep learning 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 non-trivial trade-off 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, researchers have proposed numerous solutions from the neural networks, and schedulability aspects. For example, Multi-Scale DenseNet (MSDNet) [315] and the Approximation-aware Network (APNET) [189], which we have summarized in 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/non-preemptive 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 trade-off between accuracy and inference time under schedulability constraints [1]. Thus, one of the urgent directions is to design real-time systems which include the following characteristics,

- The systems can follow the "early-exit"/"sub-network" strategies to handle the DNNs inference with time-accuracy trade-off.
- The systems are suggested to treat the scheduling (multi-task) problem as a constrained optimization problem, with overall expected inference accuracy as the objective and utilization-based schedulability tests as constraints.
- To solve the constrained optimization problem (which is probably NP-hard), the systems require the solvers, which can approximate the optimal design of the real-time system with multiple DNNs inference tasks, under specific scheduling algorithms (e.g., Earliest Deadline-First (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 back-propagation issues. Fully training a deep neural network 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 re-calculation for the back-propagation of the layers when the target application just changes a Little (e.g., form 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 trade-off 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 multi-scale multi-task 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., support vector machine 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 multi-scale dense network [191], the early-exit layers will be pre-trained in a multi-task 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 multi-task 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 paper, we introduced a survey on the state-of-the-art machine learning 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 machine learning, 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 machine learning algorithms, machine learning algorithms for scheduling problems, and security & privacy issues related to the implementation of machine algorithms. We attempted to summarize all existing machine learning 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 machine learning 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 IoTs 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*. 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," *Journal of the ACM (JACM)*, 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 Systems*, vol. 4, no. 2, pp. 145–165, 1992.

[4] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (mpsoc) technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, 2008.

[5] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

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

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

[8] N. C. Audsley, A. Burns, M. Richardson, and A. Wellings, "Deadline monotonic scheduling," 1990.

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

[10] A. Burns and R. Davis, "Mixed criticality systems-a review," *Department of Computer Science, University of York, Tech. Rep*, pp. 1–69, 2013.

[11] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, "Machine learning, neural and statistical classification," 1994.

[12] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 science and information conference*. IEEE, 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," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

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

[15] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine learning in computer vision*. Springer Science & Business Media, 2005, vol. 29.

[16] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[17] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 9–41.

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

[19] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 485–585.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[22] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *25th annual conference on neural information processing systems (NIPS 2011)*, vol. 24. Neural Information Processing Systems Foundation, 2011.

[23] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, no. 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 *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2020, pp. 15–29.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[26] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 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 *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 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," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 8, no. 04, p. 1750048, 2017.

[32] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[33] A. E. Eiben and M. Schoenauer, "Evolutionary computing," *Information Processing Letters*, vol. 82, no. 1, pp. 1–6, 2002.

[34] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2005, vol. 65.

[35] H. Kautz and B. Selman, "Pushing the envelope: Planning, propositional logic, and stochastic search," in *Proceedings of the National Conference on Artificial Intelligence*. Citeseer, 1996, pp. 1194–1201.

[36] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[37] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, 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. IEEE, 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 Transactions on Instrumentation and Measurement*, vol. 57, no. 8, pp. 1725–1733, 2008.

[40] Ş. Vădineanu and M. Nasri, "Robust and accurate period inference using regression-based techniques," in *IEEE Real-Time Systems Symposium (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 *2008 Real-Time Systems Symposium*. IEEE, 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 *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, 2018, pp. 1–6.

[43] D. Valencia and A. Alimohammad, "A real-time spike sorting system using parallel osort clustering," *IEEE transactions on biomedical circuits and systems*, 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, p. e0194889, 2018.

[45] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[46] X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou, "Interpretable deep learning: Interpretations, interpretability, trustworthiness, and beyond," 2021.

[47] G. Harman and S. Kulkarni, *Reliable reasoning: Induction and statistical learning theory*. MIT Press, 2012.

[48] L. Sha, T. Abdelzaher, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, A. K. Mok *et al.*, "Real time scheduling theory: A historical perspective," *Real-time systems*, 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 computing surveys (CSUR)*, vol. 43, no. 4, pp. 1–44, 2011.

[50] N. Shahid and S. Aneja, "Internet of things: Vision, application areas and research challenges," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2017, pp. 583–587.

[51] K. K. Patel, S. M. Patel *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 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," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[53] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[54] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[55] L. Xu and M. I. Jordan, "On convergence properties of the em algorithm for gaussian mixtures," *Neural computation*, vol. 8, no. 1, pp. 129–151, 1996.

[56] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[57] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[58] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[59] R. E. Schapire, "A brief introduction to boosting," in *Ijcai*, vol. 99. Citeseer, 1999, pp. 1401–1406.

[60] L. Devroye and T. J. Wagner, "8 nearest neighbor methods in discrimination," *Handbook of Statistics*, vol. 2, pp. 193–197, 1982.

[61] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[62] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[63] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[64] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 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," *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[67] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 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*. 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, T. Han, Q. Yang, X. Yang, Y. Wang, F. Li, and H. Wen, "Cs-cnn: Enabling robust and efficient convolutional neural networks inference for internet-of-things applications," *IEEE Access*, vol. 6, pp. 13 439–13 448, 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)," *Journal of Real-Time Image Processing*, 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 of Things Journal*, 2020.

[73] Z. Huang, X. Xu, J. Ni, H. Zhu, and C. Wang, "Multimodal representation learning for recommendation in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 675–10 685, 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*, 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 of Things Journal*, vol. 7, no. 2, pp. 1072–1080, 2019.

[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 *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

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

[78] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

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

[80] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

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

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

[83] P. Soriano, F. Caballero, A. Ollero, and C. A. de Tecnologıas Aeroespaciales, "Rf-based particle filter localization for wildlife tracking by using an uav," in *International Symposium of Robotics*, 2009.

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

[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 of Things Journal*, vol. 7, no. 7, pp. 6392–6401, 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 Transactions on Wireless Communications*, vol. 20, no. 1, pp. 375–388, 2020.

[87] H. Yang, Z. Xiong, J. Zhao, D. Niyato, Q. Wu, H. V. Poor, and M. Tornatore, "Intelligent reflecting surface assisted anti-jamming communications: A fast reinforcement learning approach," *IEEE transactions on wireless communications*, vol. 20, no. 3, pp. 1963–1974, 2020.

[88] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*. IEEE, 2007, pp. 469–475.

[89] Y. Yang, F. Nan, P. Yang, Q. Meng, Y. Xie, D. Zhang, and K. Muhammad, "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 Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[91] N. Ghourchian, M. Allegue-Martinez, and D. Precup, "Real-time indoor localization in smart homes using semi-supervised learning," in *Twenty-Ninth IAAI Conference*, 2017.

[92] J. A. Snyman, *Practical mathematical optimization*. Springer, 2005.

[93] C. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE computational intelligence magazine*, vol. 1, no. 1, pp. 28–36, 2006.

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

[95] D. P. Heyman and M. J. Sobel, *Stochastic models in operations research: stochastic optimization*. Courier Corporation, 2004, vol. 2.

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

[97] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with edf scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1250–1258, 2009.

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

[99] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, 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 *2011 IEEE 32nd Real-Time Systems Symposium*. IEEE, 2011, pp. 34–43.

[101] M. Sjodin and H. Hansson, "Improved response-time analysis calculations," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*. IEEE, 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 *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 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 *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[104] J. Y. Leung, *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press, 2004.

[105] J. J. Hopfield and D. W. Tank, ""neural" computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

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

[107] C. Cardeira and Z. Mammeri, "Neural networks for multiprocessor real-time scheduling," in *Proceedings Sixth Euromicro Workshop on Real-Time Systems*. IEEE, 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 1995*. Elsevier, 1995, pp. 67–72.

[109] C. Cardeira and Z. Mammeri, "Neural network versus max-flow algorithms for multiprocessor real-time scheduling," in *Proceedings of*

the Eighth Euromicro Workshop on Real-Time Systems. IEEE, 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 *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*. IEEE, 1991, pp. 1361–1366.

[111] M. P. Silva, C. Cardeira, and Z. Mammeri, "Solving real-time scheduling problems with hopfield-type neural networks," in *Proceedings of the 23rd EUROMICRO Conference: New Frontiers of Information Technology*. IEEE, 1997, pp. 671–678.

[112] R.-M. Chen, "Reducing network and computation complexities in neural based real-time scheduling scheme," *Applied Mathematics and Computation*, 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," *Journal of scheduling*, 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 *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021.

[115] Z. Guo and S. K. Baruah, "A neurodynamic approach for real-time scheduling via maximizing piecewise linear utility," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 2, pp. 238–248, 2015.

[116] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (can) refuted," *proc. of the 5th Int. Work. on Real-Time Net.(RTN'06)*, pp. 5–10, 2006.

[117] F. Cerqueira, F. Stutz, and B. B. Brandenburg, "Prosa: A case for readable mechanized schedulability analysis," in *Euromicro Conference on Real-Time Systems (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 *IEEE Real-Time Systems Symposium (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," *Applied Soft Computing*, 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 *Real-Time Systems Symposium (RTSS)*. IEEE, 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 *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 131–142.

[122] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The mälardalen wcet benchmarks: Past, present and future," in *10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[123] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, 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 *18th International Workshop on Worst-Case Execution Time Analysis (WCET)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[125] T. Huybrechts, T. Cassimon, S. Mercelis, and P. Hellinckx, "Introduction of deep neural network in hybrid wcet analysis," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Springer, 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, p. 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 *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, ser. OpenAccess Series in Informatics (OASIcs), J. Reineke, Ed., vol. 57. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 5:1–5:9. [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. Surv.*, vol. 52, no. 1, Feb. 2019. [Online]. Available: https://doi.org/10.1145/3301283

[129] A. Burns, "Multi-model systems — an mcs by any other name," in *8th International Workshop on Mixed Criticality Systems*, 2020.

[130] K. Agrawal, S. Baruah, and A. Burns, "Semi-clairvoyance in mixed-criticality scheduling," in *2019 IEEE Real-Time Systems Symposium (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 *IEEE Real-Time Systems Symposium (RTSS)*, 2020.

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

[133] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 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," *The journal of machine learning research*, 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?" *arXiv preprint arXiv:2003.03033*, 2020.

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

[137] Q. Xing, M. Xu, T. Li, and Z. Guan, "Early exit or not: Resource-efficient blind quality enhancement for compressed images," in *European Conference on Computer Vision*. Springer, 2020, pp. 275–292.

[138] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey, "Faster cnns with direct sparse convolutions and guided pruning," *arXiv preprint arXiv:1608.01409*, 2016.

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

[140] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695.

[141] T. Vieira and J. Eisner, "Learning to prune: Exploring the frontier of fast and accurate parsing," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 263–278, 2017.

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

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

[144] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

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

[146] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 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," *arXiv preprint arXiv:1510.00149*, 2015.

[148] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (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 *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[150] S. Ye, X. Feng, T. Zhang, X. Ma, S. Lin, Z. Li, K. Xu, W. Wen, S. Liu, J. Tang *et al.*, "Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm," *arXiv preprint arXiv:1903.09769*, 2019.

[151] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

[153] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, "Face model compression by distilling knowledge from neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[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 *Proceedings of the 13th International Conference on Web Search and 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 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 Processing Magazine*, vol. 35, no. 1, pp. 126–136, 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 420–425, 2002.

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

[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 transactions on multimedia*, vol. 7, no. 5, pp. 907–919, 2005.

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

[161] A. H. van der Heijden, *Selective attention in vision*. 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 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.

[165] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic cnn compression via low-rank decomposition with knowledge transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 2889–2905, 2018.

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

[167] X. Ruan, Y. Liu, C. Yuan, B. Li, W. Hu, Y. Li, and S. Maybank, "Edp: An efficient decomposition and pruning scheme for convolutional neural network compression," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[168] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavskỳ, V. Glukhov, I. Oseledets, and A. Cichocki, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *European Conference on Computer Vision*. Springer, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 674–10 683.

[170] B. Wu, D. Wang, G. Zhao, L. Deng, and G. Li, "Hybrid tensor decomposition in neural network compression," *Neural Networks*, vol. 132, pp. 309–320, 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 *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

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

[173] S. Lee and S. Nirjon, "Neuro. zero: a zero-energy neural network accelerator for embedded sensing and inference systems," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, 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," *The Journal of Machine Learning Research*, 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 *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4114–4122.

[176] K. Yang, T. Xing, Y. Liu, Z. Li, X. Gong, X. Chen, and D. Fang, "cdeeparch: A compact deep neural network architecture for mobile sensing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2043–2055, 2019.

[177] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 65–74.

[178] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[179] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 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, 2020.

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

[182] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 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 Geoscience and Remote Sensing Letters*, vol. 16, no. 3, pp. 412–416, 2018.

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

[185] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 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 *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 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," *arXiv preprint arXiv:2006.04152*, 2020.

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

[189] S. Bateni and C. Liu, "Apnet: Approximation-aware real-time neural network," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 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 *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 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," *arXiv preprint arXiv:1703.09844*, 2017.

[192] IntelAI, "https://www.intel.com/content/www/us/en/artificial-intelligence/posts/deep-learning-training-and-inference.html," *Blog*, 2020.

[193] E. Lind and Ä. Pantigoso Velasquez, "A performance comparison between cpu and gpu in tensorflow," 2019.

[194] S. A. Mojumder, M. S. Louis, Y. Sun, A. K. Ziabari, J. L. Abellán, J. Kim, D. Kaeli, and A. Joshi, "Profiling dnn workloads on a volta-based dgx-1 system," in *IEEE International Symposium on 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 *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 1092–1098.

[196] N. P. Toolkit, "Nvidia perfkit," 2014.

[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 *IEEE International Parallel and Distributed Processing Symposium (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 *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 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 *Proceedings of the 2018 Workshop on Mobile Edge Communications*, 2018, pp. 31–36.

[200] P. Viola, M. Jones *et al.*, "Robust real-time object detection," *International journal of computer vision*, vol. 4, no. 34-47, p. 4, 2001.

[201] A. Grubb and D. Bagnell, "Speedboost: Anytime prediction with uniform near-optimality," in *Artificial Intelligence and Statistics*. PMLR, 2012, pp. 458–466.

[202] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 572–579.

[203] K. Trapeznikov and V. Saligrama, "Supervised sequential classification under budget constraints," in *Artificial Intelligence and Statistics*. PMLR, 2013, pp. 581–589.

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

[205] Z. Xu, M. Kusner, K. Weinberger, and M. Chen, "Cost-sensitive tree of classifiers," in *International conference on machine learning*. PMLR, 2013, pp. 133–141.

[206] F. Nan, J. Wang, and V. Saligrama, "Feature-budgeted random forest," in *International conference on machine learning*. PMLR, 2015, pp. 1983–1991.

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

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

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

[210] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1039–1048.

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

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

[213] A. R. Zamir, T.-L. Wu, L. Sun, W. B. Shen, B. E. Shi, J. Malik, and S. Savarese, "Feedback networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1308–1317.

[214] E. Bini, G. Buttazzo, and G. Lipari, "Minimizing cpu energy in real-time systems with discrete speed management," *ACM Transactions on Embedded Computing Systems (TECS)*, 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 Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1871–1878, 2011.

[217] H. Hoffmann, "Coadapt: Predictable behavior for accuracy-aware applications running on power-aware systems," in *2014 26th Euromicro Conference on Real-Time Systems*. IEEE, 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 *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 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 *IEEE Real-Time Systems Symposium (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 *2016 USENIX Annual Technical Conference (USENIX ATC'16)*, 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 *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 75–86.

[223] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "Deepx: A software accelerator for low-power deep learning inference on mobile devices," in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 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 *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and 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 of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[226] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[227] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 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 *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 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 Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

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

[231] Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 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 *International ISCIS Security Workshop*. Springer, Cham, 2018, pp. 79–89.

[233] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K. R. Choo, and D. E. Newton, "Drthis: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94–104, 2019.

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

[235] A. Ruhland, C. Prehofer, and O. Horst, "embsfi: An approach for software fault isolation in embedded systems," in *Proceedings of the 1st workshop on security and dependability of critical embedded real-time systems, Porto, Portugal*, 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 *2018 International Conference on Computer and Applications (ICCA)*, 2018.

[237] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1686–1721, 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 *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 415–426.

[239] Y. Huang, H. Xiong, K. Leach, Y. Zhang, P. Chow, K. Fua, B. A. Teachman, and L. E. Barnes, "Assessing social anxiety using gps trajectories and point-of-interest data," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 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 Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, p. 39, 2015.

[241] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4w1h in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[242] J. Bian, H. Xiong, Y. Fu, and S. Das, "Cswa: Aggregation-free spatial-temporal community sensing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

[244] L. Wang, D. Zhang, A. Pathak, C. Chen, H. Xiong, D. Yang, and Y. Wang, "Ccs-ta: quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 683–694.

[245] J. Bian, H. Xiong, Y. Fu, J. Huan, and Z. Guo, "Mp2sda: Multi-party parallelized sparse discriminant learning," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 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 Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2081–2090, 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 *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 406–418.

[248] Z. Feng, H. Xiong, C. Song, S. Yang, B. Zhao, L. Wang, Z. Chen, S. Yang, L. Liu, and J. Huan, "Securegbm: Secure multi-party gradient boosting," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1312–1321.

[249] Q. Kang, J. Liu, S. Yang, H. Xiong, H. An, X. Li, Z. Feng, L. Wang, and D. Dou, "Quasi-optimal data placement for secure multi-tenant data federation on the cloud," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 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 of Things Journal*, pp. 1–1, 2021.

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

[252] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized iot data management using blockchain and trusted execution environment," in *2018 IEEE International Conference on Information Reuse and Integration (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." *Journal of Medical Systems*, 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 of Things Journal*, vol. 6, no. 1, pp. 388–398, 2019.

[255] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (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 *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018, pp. 112–117.

[257] W. Abbass, Z. Bakraouy, A. Baina, and M. Bellafkih, "Classifying iot security risks using deep learning algorithms," in *2018 6th International Conference on Wireless Networks and Mobile Communications (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 *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (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 Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[260] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020.

[261] S. M. Tahsien, H. Karimipour, and P. Spachos, "Machine learning based solutions for security of internet of things (iot): A survey," *Journal of Network and Computer Applications*, vol. 161, p. 102630, 2020.

[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," in *Internet of Things*, vol. 11, 2020, p. 100227.

[263] K. D. Ahmed and S. Askar, "Deep learning models for cyber security in iot networks: A review," *International Journal of Science and Business*, 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," *Computer Science Review*, vol. 40, p. 100389, 2021.

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

[266] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood, and Y. Xie, "Deepsniffer: A dnn model extraction framework based on learning architectural hints," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 385–399. [Online]. Available: https://doi.org/10.1145/3373376.3378460

[267] W. Y. B. Lim, Z. Xiong, J. Kang, D. Niyato, C. Leung, C. Miao, and X. Shen, "When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 457–466, 2020.

[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 of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[269] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Optimizing task assignment for reliable blockchain-empowered federated edge learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1910–1923, 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 Communications*, vol. 17, no. 9, pp. 105–118, 2020.

[271] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 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 Network*, vol. 35, no. 5, pp. 34–41, 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 Communications Surveys & Tutorials*, 2021.

[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 *TENCON 2008-2008 IEEE region 10 conference*. IEEE, 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," *Engineering Applications of Artificial Intelligence*, vol. 96, p. 104000, 2020.

[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 of Things Journal*, vol. 8, no. 5, pp. 3170–3179, 2020.

[277] N. Dey, S. Fong, W. Song, and K. Cho, "Forecasting energy consumption from smart home sensor network by deep learning," in *International conference on smart trends for information technology and computer communications*. Springer, 2017, pp. 255–265.

[278] S. A. Boyer, *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

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

[280] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and reliability engineering international*, 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," *Mechanical Systems and Signal Processing*, 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 *2019 Chinese Automation Congress (CAC)*. IEEE, 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," *arXiv preprint arXiv:1911.05376*, 2019.

[284] P. Ferrari, S. Rinaldi, E. Sisinni, F. Colombo, F. Ghelfi, D. Maffei, and M. Malara, "Performance evaluation of full-cloud and edge-cloud architectures for industrial iot anomaly detection based on deep learning," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*. IEEE, 2019, pp. 420–425.

[285] I. Lawrence and K. Lin, "Assay validation using the concordance correlation coefficient," *Biometrics*, pp. 599–604, 1992.

[286] H.-H. Hsu and C.-C. Chen, "Rfid-based human behavior modeling and anomaly detection for elderly care," *Mobile Information Systems*, 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 *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 *2nd International Conference on Data, Engineering and Applications (IDEA)*. IEEE, 2020, pp. 1–7.

[289] F. Ali, S. El-Sappagh, S. R. Islam, D. Kwak, A. Ali, M. Imran, and K.-S. Kwak, "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Information Fusion*, vol. 63, pp. 208–222, 2020.

[290] X. Wu, C. Liu, L. Wang, and M. Bilal, "Internet of things-enabled real-time health monitoring system using deep learning," *Neural Computing and Applications*, pp. 1–12, 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 *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2017, pp. 78–81.

[292] S. Tennyson and P. Salsas-Forn, "Claims auditing in automobile insurance: fraud detection and deterrence objectives," *Journal of Risk and 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," *Journal of Risk and 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," *Journal of Risk and 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," *Decision Support Systems*, vol. 105, pp. 87–95, 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 *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*. IEEE, 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 *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1866–1875.

[298] E. Priyanka, M. G. Shankar, S. Tharun, S. Ravisankar, S. N. Saravanan, B. B. Kumar, and C. Pugazhenthi, "Real-time performance analysis of multiple parameters of automotive sensor's can data to predict vehicle driving efficiency," *International Journal of Computing and Digital System*, 2021.

[299] S. P. Sethi, H. Yan, and H. Zhang, *Inventory and supply chain management with forecast updates*. Springer Science & Business Media, 2005, vol. 81.

[300] Y. M. Lee, F. Cheng, and Y. T. Leung, "Exploring the impact of rfid on supply chain dynamics," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*, vol. 2. IEEE, 2004, pp. 1145–1152.

[301] B. Sahay and J. Ranjan, "Real time business intelligence in supply chain analytics," *Information Management & Computer Security*, 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," *Journal of Self-Governance and Management Economics*, 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*. Springer, 2020, pp. 537–547.

[304] R. Gonzalez, F. Manco, A. Garcia-Duran, J. Mendes, F. Huici, S. Niccolini, and M. Niepert, "Net2vec: Deep learning for the network," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2017, pp. 13–18.

[305] N. Kargah-Ostadi, A. Waqar, and A. Hanif, "Automated real-time roadway asset inventory using artificial intelligence," *Transportation Research Record*, 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 Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 63–77, 2006.

[307] Z. Zhu and Q. Ji, "Real time and non-intrusive driver fatigue monitoring," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*. IEEE, 2004, pp. 657–662.

[308] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-time imaging*, 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," *Journal of Intelligent & Robotic Systems*, 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 Intelligent Transport Systems*, 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," *Transportation research record*, 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," *Journal of Real-Time Image Processing*, 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 *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020, pp. 205–217.

[314] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[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 *International Conference on Learning Representations*, 2018.

[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," *OSPERT'16*, 2016.

[317] H. Zhou, S. Bateni, and C. Liu, "Sˆ 3dnn: Supervised streaming and scheduling for gpu-accelerated real-time dnn workloads," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 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 *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 104–115.

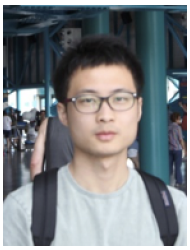
[319] Y. Xiang and H. Kim, "Pipelined data-parallel cpu/gpu scheduling for multi-dnn real-time inference," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 392–405.

[320] Y. Bao, Y. Peng, Y. Chen, and C. Wu, "Preemptive all-reduce scheduling for expediting distributed dnn training," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 626–635.

[321] Y. Choi and M. Rhu, "Prema: A predictive multi-task scheduling algorithm for preemptible neural processing units," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 220–233.

[322] O. Wechsler, M. Behar, and B. Daga, "Spring hill (nnp-i 1000) intel's data center inference chip," in *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 2019, pp. 1–12.

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

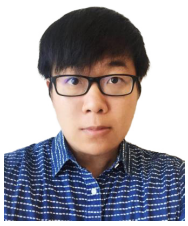


Jiang Bian (Member, IEEE) received the Ph.D. degree of Computer Engineering in University of Central Florida, Orlando, FL. He received the B.Eng degree of Logistics Systems Engineering in Huazhong University of Science and Technology, Wuhan, China, in 2014, and the M.Sc degree of Industrial Systems Engineering in University of Florida at Gainesville, FL, in 2016. He is currently with the Big Data Laboratory, Baidu Research, Beijing, China. His research interests include Ubiquitous Computing, and AutoDL.

Abdullah Al Arafat (Student Member, IEEE) is currently pursuing his Ph.D. in Computer Engineering at the University of Central Florida (UCF), Orlando, Florida, USA. He received the BS in Electrical Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 2016 and MS in Computer Engineering from UCF in 2020. His research interests include Scheduling Theory, Algorithms, and Real-Time & 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 and a Postdoc at University of Virginia, Charlottesville, VA from 2015 to 2016. He is currently a Principal Research Scientist at Big Data Lab, Baidu Research, Beijing, China, and also a Graduate Faculty Scholar affiliated to University of Central Florida, Orlando FL. His current research interests include AutoDL and ubiquitous computing. He has published more than 70 papers in top computer science conferences and journals.

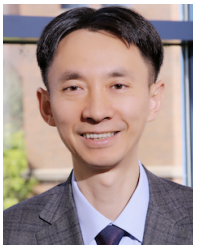
Jing Li (Member, IEEE) is an Assistant Professor with the Department of Computer Science, New Jersey Institute of Technology. She received her Ph.D. at Washington University in St. Louis in 2017, where she was advised by Professor Chenyang Lu and Kunal Agrawal. She received B.S. in computer science from Harbin Institute of Technology in 2011. Her research interests include real-time systems, parallel computing, cyber-physical systems, and reinforcement learning for system design and optimization. She has high impact publications in top journals and conferences with 3 outstanding paper awards.



Li Li (Member, IEEE) is currently an assistant professor in the University of Macau. He received his Ph.D. degree and M.S. degree in Electrical and Computer Engineering from Ohio State University, Columbus, OH, USA in 2018 and 2014 respectively. He obtained the B.S. degree from Tianjin University in 2011. He has published in refereed journals and conference proceedings, such as INFOCOM, RTSS, ICDCS, NDSS, MM, TMC, TDSC.

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, China. He is currently a Senior Research Expert with Zhejiang Laboratory, China. 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) is a Professor of Computer Engineering; and Director of the Computer Architecture and Storage Systems (CASS) Laboratory at the University of Central Florida, Orlando, FL, USA. He has conducted extensive research in the areas of Computer Systems and Data-Intensive Computing. 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. He has authored over 150 publications in premier journals and conferences.



Dejing Dou (Senior Member, IEEE) is the Head of Big Data Lab (BDL) and Business Intelligence Lab (BIL) at Baidu Research. He is also a full Professor (on leave) from the Computer and Information Science Department at the University of Oregon and has led the Advanced Integration and Mining (AIM) Lab since 2005. He has been the Director of the NSF IUCRC Center for Big Learning (CBL) since 2018. He was a visiting associate Professor at Stanford Center for Biomedical Informatics Research during 2012-2013. Prof. Dou received his bachelor degree from Tsinghua University, China in 1996 and his Ph.D. degree from Yale University in 2004. His research areas include artificial intelligence, data mining, data integration, NLP, and health informatics. Dejing Dou has published more than 100 research papers.

Zhishan Guo (Senior Member, IEEE) is an Assistant Professor with the Department of Computer and Electrical Engineering, University of Central Florida, Orlando, FL, USA. He received the B.Eng. degree (with honor) 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. His current research interests include real-time and cyber-physical systems, neural networks, and computational intelligence. He has received best paper awards from flagship conferences such as RTSS and EMSOFT.