Received 13 November 2022, accepted 16 November 2022, date of publication 28 November 2022, date of current version 2 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225092

SURVEY

Deep Learning on Energy Harvesting IoT Devices: Survey and Future Challenges

MINGSONG LV, (Member, IEEE), AND ENYU XU
International Laboratory for Smart Systems, Northeastern University, Shenyang 110819, China

Corresponding author: Mingsong Lv (lumingsong@cse.neu.edu.cn)

This work was supported in part by the NSFC Project under Grant 61772123.

ABSTRACT Internet-of-Things (IoT) devices are becoming both intelligent and green. On the one hand, Deep Neural Network (DNN) compression techniques make it possible to run deep learning applications on devices equipped with low-end microcontrollers (MCUs). By performing deep learning on IoT devices, in-situ decision-making can be made, which can improve the responsiveness of such devices to the environment and reduce data uploading to edge servers or clouds to save valuable network bandwidth. On the other hand, many IoT devices in the future will be powered by energy harvesters instead of batteries to reduce environmental pollution and achieve permanent service free of battery maintenance. As the energy output of energy harvesters is tiny and unstable, energy harvesting IoT (EH-IoT) devices will experience frequent power failures during their execution, making the software task hard to progress. The deep learning tasks running on such devices must face this challenge and, at the same time, ensure satisfactory execution efficiency. We believe deploying deep learning on EH-IoT devices that execute intermittently will be a challenging yet promising research direction. To motivate research in this direction, this paper summarizes existing solutions and provides an in-depth discussion of future challenges that deserve further investigation. With IoT devices becoming more intelligent and green, DNN inference on EH-IoT devices will generate a much more significant impact in the future in academia and industry.

INDEX TERMS DNN inference, energy harvesting, IoT devices, embedded systems.

I. INTRODUCTION

Energy harvesting [1] is believed to be the future technique to power IoT devices for two main concerns: first, millions of IoT devices will be deployed in hard-to-reach working environments, such as mountains and forests, charging and replacing the batteries of such devices will be almost impossible. Energy harvesting will eliminate battery maintenance and significantly extend the lifetime of the devices beyond the limit of battery size. Second, batteries on such devices are identified as a significant source of environmental pollution since the batteries in the vast amount of tiny IoT devices can hardly be reclaimed after deployment. By replacing batteries with energy harvesters, the threat of IoT devices to the natural environment is considerably reduced.

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

A significant problem of energy harvesting is that the energy output of the harvesters is typically tiny and unstable due to uncertainty in the energy harvesting environment. Take solar energy harvesting, for example, the availability of the energy depends on a broad range of factors, such as sunlight angle, season, the weather of the day, etc. The change of any factor will significantly affect the energy harvesting power. The energy input is even more unpredictable for other sources, such as wind and thermal energy. As a result, the devices will experience frequent power failures during their execution. When a power failure occurs, the running program will be stopped, and the computation context in the main memory will be lost. After energy regains and the system restarts, the program must restart from the beginning and re-execute what has already been done before the power failure. In the extreme case, the program may never make any progress if power failures occur very frequently [1].

Intermittent computing [2] aims to enable a program to make incremental progress across power failures. The main technique is to checkpoint the execution states (registers, memory contents) of a program to non-volatile memory (NVM) before a power failure occurs and to reload the checkpointed states after energy regains so that the program can resume from where it was left [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Besides checkpointing-based approaches, task-based approaches [14], [15], [16], [17], [18], [19] allow programmers to choose the program points for checkpointing and the data contents to be saved. Typically, a programmer needs to write computation logic into software segments called tasks and explicitly specify what data or variables should be saved to NVM when the current task is finished. For both the above techniques, a major problem is that checkpointing the execution states to NVM incurs significant time and energy overhead, which will severely impede the progress of the software task.

Another trend is that IoT devices are becoming more intelligent. Deep learning, specifically deep neural networks, was traditionally carried out on powerful GPUs or data centers as they require a considerable amount of computation [20]. Recently, new DNNs with small footprints were developed to run on MCUs with very limited resource [21]. Such DNNs may have from several megabytes to hundreds of megabytes of memory consumption. The rationale for running DNNs on MCUs is that sensor data can be analyzed in situ on the IoT device. The benefits are two-fold: first, by doing data analysis on the device side, decision-making can be performed in real-time so that the device may have better responsiveness to the environment; second, data transfer between IoT devices and the cloud will be significantly reduced, thus saving valuable network bandwidth.

We believe the development in the above two trends is worth special attention since they will make IoT devices both mentally and physically ready for future application scenarios. "Mentally ready" means it is possible to run AI applications on IoT devices for high-quality analysis and decision-making. "Physically ready" means the devices should be appropriately powered for long-time service, free of battery size limitations. Deep learning and energy harvesting are the two pillars to make IoT devices ready for their future form. In a recently proposed concept "WEAF Mnecosystem" [22], [23], AI and energy are identified as the two critical enablers for the self-evolution of future IoT systems.

The merging of the two trends has created a new scenario of IoT computation: to perform deep learning tasks on intermittently powered EH-IoT devices [24]. Researchers are exploring solutions enabling intermittent deep learning on EH-IoT devices. Unlike the simple control programs discussed in existing intermittent computing research [25], Deep learning programs typically generate large feature maps as intermediate computation data before a final classification/identification result is produced. To enable such programs to progress across frequent power failures, the feature maps will be checkpointed to NVM as part of the program execution states. This will incur a much more significant overhead compared to simple control programs. Reducing this overhead to improve inference efficiency in the energy harvesting environment remains largely unsolved. Although intermittent deep learning has only been researched for about five years, we believe this will become a critical computation technology for future IoT devices and significantly impact academia and industry.

This paper aims to pinpoint the evolving research on intermittent deep learning by reviewing current research work and solutions and then performing an in-depth discussion on the future challenges requiring special attention to push state-of-the-art intermittent deep learning. Note that this paper is not intended to provide comprehensive surveys on energy harvesting technology, intermittent computing, or deep learning. Extensive survey work has been done on the above research areas [1], [20], [21], [25], which interested readers can reference. In Section II, we briefly provide a glimpse of the techniques in the above research areas to provide the background knowledge for better understanding.

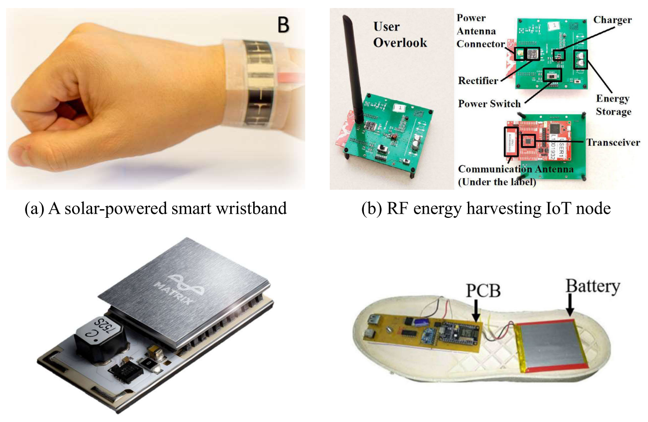
The rest of the paper is organized as follows. In Section II, we present some background knowledge needed to understand the technical contents, including energy harvesting IoT devices, intermittent computing, and deep neural networks. The review of existing research on intermittent deep learning is given in Section III by focusing on different technical problems. The discussion on future research is provided in Section IV, and the paper is concluded in Section V.

II. BACKGROUND KNOWLEDGE

This section provides essential knowledge on energy-harvesting IoT devices, intermittent computing, and DNN.

A. ENERGY HARVESTING IoT

IoT devices are generally designed as small devices equipped with sensors, MCU, and wireless networking capability. Such devices are expected to be deployed in various working environments where a wired power supply is generally impossible. Therefore, the devices are powered by on-device power sources currently realized by various types of batteries. However, *battery technology has several major limitations* [26]. First, batteries need to be recharged or replaced when the energy inside the battery is depleted, which is inconvenient or even impossible for devices deployed in harsh working environments, such as fire-warning IoT devices scattered in the mountains. To prolong the battery life, IoT devices are typically equipped with low-power hardware components and are configured to be activated very infrequently to save energy. One example is the heart pacemaker. A pacemaker implanted in human hearts can work for about seven years. Even though, when the pacemaker's battery is depleted, the patient must experience a heart operation to change the pacemaker's battery, which is very inconvenient and costly [27]. Second, batteries, especially those carried on millions or billions of IoT devices, represent a significant environmental threat because the batteries can hardly be reclaimed from the harsh working environment. The above



(a) A solar-powered smart wristband (b) RF energy harvesting IoT node

(c) A thermal energy harvesting module (d) A kinetic energy harvesting shoe

FIGURE 1. Examples of energy harvesting techniques. (a) a solar-powered smart wristband [29]; (b) an RF energy harvesting IoT node [30]; (c) a thermal energy harvesting module by MATRIX [31]; (d) a kinetic energy harvesting shoe [32].

two significant problems are now hindering the development of IoT.

In recent years, energy harvesting is becoming popular as an alternative energy source for IoT devices, which is very promising for solving battery problems. In this paper, we call IoT devices powered by energy harvesting EH-IoT devices. The main component of EH-IoT devices is the energy harvester which collects energy from the ambient environment. There are several types of energy harvesters [28]. Examples are given in Figure 1.

Solar energy harvesting is the most popular approach [33]. Solar panels use sunlight as a source of energy to generate direct current electricity through the photovoltaic effect. Most modules use wafer-based crystalline silicon cells or thin-film cells. The power output of a solar panel can be affected by multiple factors. First, the light intensity directly affects the power output. For example, in direct sunshine, a solar panel may work at peak power output, and in a dark environment, a solar panel has no power generation. Second, the incident light angle is also a determining factor. A solar panel generates higher power output when the panel's surface is vertical to the incident light. Third, the power output of a solar panel also depends on other factors, including temperature and payload. Traditionally, large solar panel arrays are primarily deployed in areas with sufficient sunshine, such as deserts, to generate energy that cities will use. Recently, small solar panels have been introduced to mobile IoT devices as power sources. For example, the self-powered PowerWatch [34] uses a solar panel as one of its two primary power sources.

Another approach is *radio frequency (RF) energy harvesting* [35], [36], [37]. Conventionally, IoT devices are equipped with an antenna to communicate with other devices or wireless control centers using an RF signal. RF energy harvesting treats the electromagnetic radiation scattered in the environment as a natural source of energy and converts part of such energy into electrical energy to power the IoT device. Another related technology is wireless power transfer [38], [39], by which RF energy is intentionally transmitted from a source to a target in order for the receiver to gather proper energy to support its operation. A successful platform is the WISP device [40] that harvests energy from the Radio Frequency Identification (RFID) signal. Related devices based on WISP have been used in indoor object tracking [41]. The power output of the RF energy harvester depends on many factors, including the efficiency of the antenna, the distance between the signal sender and receiver, the accuracy of the impedance matching between the antenna and the voltage multiplier, and the power efficiency of the voltage multiplier that converts the received RF signals to DC voltage.

Kinetic energy harvesting is popular in both factories and wearable IoT devices [42]. Kinetic energy harvesting is the process of converting environmental kinetic or vibration energy into electrical energy. The harvesting device is made from piezoelectric materials that work with the piezoelectric effect to convert mechanical strain into electric current or voltage. The strain can be generated from various sources, including human motion, low-frequency seismic vibrations, acoustic noise, wind, and tide. Kinetic energy is widely available on mobile devices. In [43], a shoe-based battery-free wearable sensing platform was proposed, on which the power is generated from the two feet when people walk. The designed piezoelectric energy harvester achieved a power output of $1 \sim 2$mW that is enough to power the sensors, the MCU, and the radio with reasonable task frequency.

For wearable IoT devices, *thermal energy harvesting* is a feasible and promising approach [44]. The thermal energy harvester is essentially a thermoelectric generator (TEG) that is a solid-state device that converts heat flux (temperature differences) directly into electrical energy through a phenomenon called the Seebeck effect (a form of thermoelectric effect). For example, besides a solar panel, the self-powered PowerWatch [34] is also equipped with a TEG as a second source of energy harvesting to power the smartwatch.

New materials and technologies are emerging for energy harvesting. In [45], a sensing device combines flexible polyvinylidene fluoride (PVDF) piezoelectric film with kresling origami structure to design a new piezoelectric kresling origami generator, which can achieve high-efficiency and broadband energy harvesting performance. In [46], multi-scale metamaterials with super-normal functions on energy manipulation are utilized in multi-field renewable energy harvesting and absorbing, which can enhance the local energy density by confining and focusing the energy before it is harvested. To waive the energy discontinuity problem, the literature investigates the development of multi-source EH platforms, where miniaturized energy harvesters addressing different environmental sources are integrated together [47], [48], [49]. Relying on orthogonal sources, those platforms can effectively overcome discontinuity due to the shortage of a specific source at a particular moment. These types of solutions are made particularly feasible thanks to the exploitation of Micro/Nanotechnologies and advanced packaging and integration solutions.

Energy harvesting is increasingly adopted in smart factory [50], smart transportation [51], smart building [52], implantable medical devices [53], wearable devices [54], [55], [56], etc. To enable interoperability between products from different manufacturers, the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) have initiated the standardization of EH-IoTs, including wireless protocol design and testing for different types of energy harvesting approaches [57].[1] Energy harvesting is becoming an important infrastructure for future IoT devices and applications.

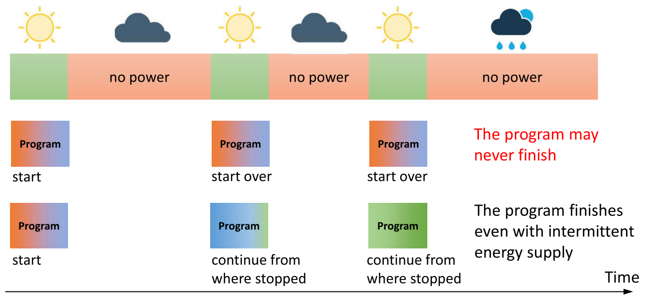

no power
no power
no power
Program
start
Program
start over
Program
start over
The program may never finish
Program
start
Program
continue from where stopped
Program
continue from where stopped
The program finishes even with intermittent energy supply
Time

FIGURE 2. An illustration of the problem of program execution in the presence of frequent power failures and the basic idea of intermittent computing.

B. INTERMITTENT COMPUTING

As can be seen from the above introduction of energy harvesting technologies, the run-time power output of the energy harvesters, regardless of the specific technology, can be affected by many factors and thus may be very unpredictable. For example, the power output of a solar panel is very sensitive to the weather and whether the panel is directly radiated by the light source [58], [59]. Consider a solar-powered smartwatch user; the watch's panel may generate enough energy in the daytime with sufficient sunlight. Still, the panel may generate almost zero output on cloudy days or in dark environments. When the user is walking, the plane direction of the solar panel will change all the time, which will also considerably reduce its power output. Such problems are even more prominent in IoT devices, the application environment of which can be very unpredictable.

The unpredictable output of energy harvesters will lead to a severe issue: *the EH-IoT device will experience frequent power failures during its execution*. Typically, the execution of a software program is based on the program states in volatile memory, such as the variables stored in registers and main memory. Once a power failure occurs, the program states in the volatile memory will be lost. As a result, when the energy regains and the system restarts, the program must re-execute from the very beginning. This will cause the program hard to progress. In the worst case, if the power failure occurs too frequently, the program may have no chance to progress to the end [1]. The problem is shown by Figure 2.

Intermittent computing [2] was proposed to solve the above problem. The main idea is to enable a software program to progress incrementally by frequently checkpointing the program states to NVM. Once a power failure occurs, the program can restart from the latest checkpoint by reloading the checkpointed program states. Thus, it does not have to restart from the beginning (i.e., achieving incremental progress).

There are two typical techniques to achieve intermittent computing: *checkpointing-based approach and task-based approach*. Checkpointing is a technique widely used in fault-tolerant computing. When program execution reaches a checkpoint, the system states will be copied from the volatile main memory to the NVM, where the program can be resumed. This technique was widely used in intermittent computing [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [60]. The checkpointing technique is program-friendly since it typically does not require the programmer to specify the specific checkpointed data. However, checkpointing is generally expensive, as the checkpointing system typically saves a large set of system states to NVM.

The *task-based approach* [14], [15], [16], [17], [18], [19] offers a lighter-weight alternative to checkpointing. Take the task-based intermittent computing systems InK [16] for example, an application is programmed as a collection of atomic execution blocks called *tasks*. Each task is written as a function, and the programmer explicitly defines the control flow between tasks. Program states are saved at task boundaries, and when recovering from a power failure that occurred during the execution of a task, the system resumes execution from the beginning of that task. At task boundaries, the program states (i.e., a subset of the program variables) to be saved to NVM (and will be restored after a power failure) will be determined by the programmer at the design time. Compared to checkpointing-based approaches, task-based approaches typically require much fewer data to be saved at task boundaries. However, the main difficulty is that the program designer must specify the data saved at task boundaries.

Nowadays, NVM is being used as the main memory in some MCUs. For example, some of the TI MSP430 MCUs [61] are using FRAM as the main memory. Therefore, the program/system states will be directly stored on NVM and will not be lost during power failures. However, directly operating data on NVM may cause memory inconsistency due to the so-called *Write-After-Read (WAR)* problem [4] in the presence of power failures. The system should maintain a backup version for the WAR variables to solve the problem. When recovering from a power failure that occurred in some tasks, the system restores the WAR variables from the backup version to the main memory. It then correctly resumes execution from the beginning of the aborted task. Some existing

[1]The IEC 62830 standard has eight parts, here we only include the reference for the first part, and the other parts can be retrieved on the IEC website.

intermittent computing systems are built on FRAM-based MCUs. But MCUs equipped with FRAM generally cost much higher than traditional MCUs equipped with SRAM and FLASH, so devices using such MCUs only occupy a small portion of the IoT devices in the literature.

For both checkpointing-based and task-based approaches, saving system/program states represent a major run-time overhead since checkpointing the states to NVM is time-consuming and energy-consuming. In some systems [16], the time overhead of checkpointing may occupy up to 90% of the total execution time. Therefore, reducing the checkpointing overhead is one of the major concerns and optimization objectives in intermittent computing.

Furthermore, IoT devices typically interact with their environment through peripherals. Maintaining consistent states across power failures is another critical problem for such systems. Without persistent peripheral states, a program may behave incorrectly in many aspects. For example, a program may get stuck in an infinite loop of state polling due to power failures [11]. Re-operation of peripherals across power failures may produce inconsistent system states [62]. Interrupts may also violate the atomicity of the program [18].

C. DEEP NEURAL NETWORKS

Machine learning is valuable in many applications, including image classification, object detection, multimedia retrieval, recommendation, social network analysis, etc. With the development of the Internet and pervasive computing, massive sensing data are generated every day, which triggers the thriving of the data-driven machine learning techniques, among which deep neural network is the most successful one [20].

DNN is a general concept covering many different neural network models. A DNN takes the sensor data as input (such as images containing objects to be identified) and processes the input data through several layers of computation. Conceptually, each layer will extract many abstract features based on the previous layer's output. In the end, the output from the last feature-extraction layer will be put into a particular layer called a classifier that generates the final identification result, e.g., the classification of the objects in an image. The output data of each feature-extraction layer is typically called *feature map*. In many successful neural network models, a larger number of layers leads to higher analysis precision. Thus, these network models are called *deep* neural networks. A DNN is effective only when it is intensively trained, i.e., a massive amount of data goes through the network many times to adjust the network parameters, which is very time-consuming. When a well-trained DNN is deployed to generate outputs, the input data will go through the network only once, and the process is called *inference*.

Currently, there are two dominating types of DNNs: convolution neural network (CNN) [63] and recurrent neural network (RNN) [64]. The structure of CNNs is inspired by the neurons in animal and human brains as it simulates the brain's visual cortex. CNN has been successfully applied in applications such as Natural Language Processing (NLP), speech processing, and computer vision. RNN is primarily used in speech processing and typically has excellent performance. In applications such as speech processing, a word is only precisely understood when the context can be considered. An RNN utilizes the sequential information in the network and thus achieves short-term memory capability. Computational-wise, a CNN can be viewed as a linear data flow model, while an RNN contains complex data dependency between different layers.

Regardless of its type, the execution of a DNN generally requires large CPU and memory usage. The high computation requirement is acceptable when the DNN is executed on a powerful computer server. With the rapid development of IoT, DNNs are increasingly adopted on IoT devices. The rationale is that application responsiveness can be improved by analyzing the sensor data on the IoT device (so that sensor data are not needed to be uploaded to a remote server for processing). However, an IoT device has very limited computation capability and memory space. For example, many IoT devices are only equipped with low-end MCUs. Such devices cannot meet the requirements of running normal DNNs. Thus, much research has been done to reduce the size of a DNN or to execute part of the DNN so that the requirement on computation and memory is reduced [21], [65]. There are several techniques to solve this problem.

The first class of work is lightweight network design. With the model size as a critical design parameter, many DNNs have been proposed with a network model size of several megabytes or hundreds of kilobytes. SqueezeNet [66] is an early network designed to run on resource-constrained hardware. It introduces a fire module in which convolution operations are split into the squeeze and expand layers. MobileNet [67], [68], [69], which has developed across many generations, is another representative lightweight DNN. The depth-wise separable convolution technique significantly reduces the complexity of the DNN. ShuffleNet [70], [71] uses the group convolution and channel shuffle techniques to reduce its footprint. These lightweight DNNs have demonstrated an accuracy very close to complex DNNs with much low computation and memory requirements in specific domains such as object detection.

The second class of work is network compression. The basic idea is to remove the redundant parts of an over-parameterized network to produce a much smaller DNN with similar analysis accuracy. Three types of techniques are developed for network compression. Network pruning [72] removes the redundant weights/channels of a complex DNN. Quantization [73] uses fewer bits to store the weight parameters and the intermediate feature maps. Knowledge distillation [74] learns a smaller DNN model from a complex DNN model. The new resulting DNNs are typically retrained to provide an analysis accuracy close to the complex DNNs while keeping their footprints at several megabytes.

The third class of work is adaptive DNN inference which allows an inference task to execute only part of the network (using only a portion of the filters or skipping some DNN layers) if the computation capability or the available energy does not allow complete execution of the whole network. Analysis precision is traded off for computation/energy efficiency. There are three main types of techniques to do adaptive inference: 1) multi-model selection [75], [76]; 2) runtime filter pruning [77], [78], [79], [80]; 3) early-exit networks [81], [82], [83], [84]. Multi-model selection requires several DNNs to be available on the device, and before the inference starts, the system will select one of the networks to execute. Filter pruning allows skipping some filters at runtime to reduce computation. Early-exit networks attach multiple classifiers at different convolution layers. When a decision is made to reduce the computation workload, the inference task may exit at an earlier point, i.e., skipping the layers after the selected point.

D. DNN INFERENCE MEETS INTERMITTENT COMPUTING

Although the techniques mentioned above have successfully reduced the sizes of DNNs, executing such DNNs on intermittently powered EH-IoT devices remains a big challenge. During the inference of a lightweight DNN, a large volume of intermediate feature data will be generated. To enable a DNN to make progress across frequent power failures, these feature maps will be checkpointed to NVM. In terms of energy consumption and execution time, the checkpointing overhead is still unacceptable even for very small DNNs because the energy input is too tiny. This challenge has attracted many researchers in recent years (which will be reviewed in Section III) and deserves further development (which will be discussed in Section IV).

III. A REVIEW OF EXISTING RESEARCH ON INTERMITTENT DNN INFERENCE

A. METHODOLOGY

Although comprehensive survey work has been done on the hot research areas, including energy harvesting technology, intermittent computing, and deep learning, this paper focuses on a new research direction, ‘‘deep learning on energy-harvesting IoT devices,’’ which is an interesting combination of the above directions to build green and intelligent IoT systems that will evolve in the future. Therefore, in surveying the state-of-the-art work, we exclude the papers that only deal with one of the above three directions. Specifically, this survey focuses on a critical issue of deploying computation-intensive DNNs on intermittently powered IoT devices. Thus, related research on using DNNs to optimize the efficiency of energy harvesting or developing better DNNs to solve a specific application target is also outside the scope of this survey and is not included. The new emerging research direction was first formally discussed in [24], but related work has been done since 2017. The literature was reviewed from Google Scholar and DBLP with an intensive search with related keywords. The closely related papers from the year 2017 to year 2022 were selected, and their contributions are detailed in Section III.

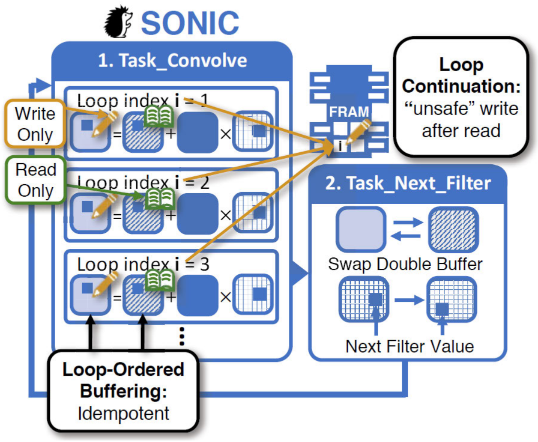


FIGURE 3. The architecture of SONIC [24].

Existing research works try to tackle the problem of intermittent DNN inference from different view angles. Related research practices are summarized in Table 1, and the survey of technical contents is given in the rest sub-sections.

B. SAFE CHECKPOINTING TECHNIQUES

The fundamental technique required for intermittent execution is checkpointing, which saves the program states to NVM frequently so that the program can make incremental progress. A DNN inference program is essentially an ordinary software program, so generally, the checkpointing techniques developed in the intermittent computing domain can also be applied to DNN inference programs. The primary concern is the overhead of checkpointing.

Gobieski, Lucia, and Beckmann are the first to present a comprehensive solution for intermittent DNN inference on energy-harvesting devices [24]. The authors proposed a runtime system called SONIC modified from the Alpaca runtime system [15], the functionality of which is to program and run a DNN inference program as a set of tasks (the so-called ‘‘task-based intermittent approach’’) across power failures. The architecture of SONIC is shown in Figure 4.

A DNN inference program is essentially a nested loop with the filter operation as the computation unit. SONIC implements a functionality called *loop continuation* to ensure incremental progress across power failures. Loop continuation stores the loop control variables and data manipulated directly in NVM (specifically a FRAM). When a loop continuation task restarts, the local volatile variables are reinitialized by reading back the loop states from the last attempted iteration.

However, only saving loop states to NVM does not guarantee correctness when the loop continuation task re-executes due to the well-known ‘‘Write After Read (WAR)’’ problem. SONIC uses a double-buffering approach to solve the problem. The input feature map is stored in one buffer for each filter operation, and the output feature map is stored in a second buffer instead of updating the input matrix.

TABLE 1. A summary of the surveyed work.

Related work	Work target	Technique	Section
Gobieski et al. [24], 2019	Correctly checkpointing a software DNN inference task	Correctly maintaining loop indices	Sec. III-B
Kang et al. [85], 2020		Correctly maintaining the footprint of the progress of inferences with hardware accelerators	
Kang et al. [86], 2022	Effectively tracking inference progress	Model augmentation to piggyback progress information into an augmented DNN	
Montanari et al. [87], 2020	Adapting the DNN inference workload to the energy envelop	Adapt input resolution according to the availability of energy; multi-exit DNN	Sec. III-C
Wu, Li, et al. [83], [84], [88], 2020-2022		DNN model compression and multi-exit DNN	
Zoph et al. [89], 2017	Designing new DNNs that are suitable for EH-IoT devices	Automatic network search incorporating energy consumption constraints	Sec. III-D
Lv et al. [90], 2022	Reducing DNN checkpointing overhead	Depth-first DNN inference to reduce the amount of feature map data to be checkpointed	Sec. III-E
Islam et al. [91], 2020	Meeting timing requirements for DNN inference on EH-IoT devices	Dynamic priority real-time scheduling + multi-exit DNN model	Sec. III-F
Resch et al. [92], [93], 2020-2022	Hardware design to facilitate checkpointing DNN inference tasks	New memory system to facilitate checkpointing	Sec. III-G
Qiu et al. [94], 2021		Run-time workload and energy monitoring + new memory system	
Lee et al. [95], 2019	Learning with intermittent power supply	Action planning + learning example selection	Sec. III-H
Islam et al. [96], 2022		Resource-aware DNN training with architecture search, model compression, normalization, and fixed-point calculation	

This approach removes WAR operations. After applying a single filter, SONIC swaps the input buffer with the output buffer and moves on to the next filter value. But this buffering approach has a performance problem: unmodified activation data will be copied between the two buffers, spending considerable time and energy. To remove this overhead, SONIC introduces *sparse undo-logging*, which tracks the program's progress via a read index and a write index. When applying a filter, SONIC copies the unmodified activation into a canonical memory location and increments the read index. Then, SONIC computes the modified activation and writes it back to the original activation buffer (no separate output buffer in this technique). After that, SONIC increments the write index and proceeds to the next iteration. However, sparse undo-logging doubles the number of memory writes per modified element, so it is inefficient on convolution layers where most data are modified. Sparse undo-logging is only used on fully connected layers to enable efficient intermittent execution.

For high performance and low power consumption, hardware accelerators are increasingly used in small, embedded devices to perform the basic mathematical computation steps in the DNN inference. The energy consumption can be reduced with hardware accelerators compared to pure software implementation, but it is still orders of magnitude higher than the energy input from the energy harvesters.

Kang et al. proposed a system called HAWAII [85] to enable intermittent DNN inference that adopts hardware accelerators. A specific challenge is that the internal state of an accelerator is typically inaccessible, which leads to the loss of inference progress and incorrect execution behavior. The authors proposed a concept called "footprinting" that preserves the state of hardware-accelerated intermittent DNN

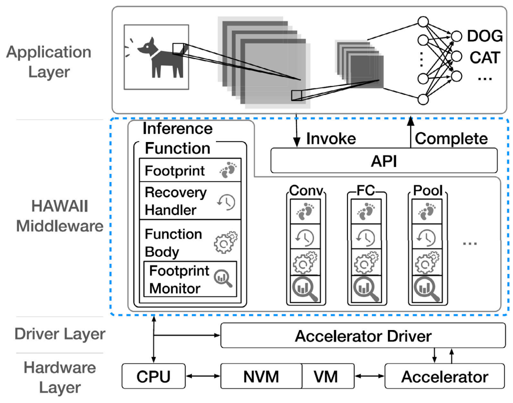


FIGURE 4. The system architecture of HAWAII [85].

inference. For most DNNs, each layer consists of neurons, essentially a set of sequentially executed sub-operations. Thus, the number of completed sub-operations indirectly represents the inference progress, i.e., the footprint of the inference program. Specifically, the footprint contains two numbers: one indicates which layer has been finished, and the other indicates which sub-operation in the layer is just completed.

To preserve inference progress across power failures, HAWAII only saves the footprints and the intermediate sub-operation results to NVM. Therefore, a sub-operation is an atomic execution unit of the intermittent inference. Since the hardware accelerator is invoked inside a sub-operation, there is no need to identify and save the states internal to the hardware accelerator. A footprint monitor is designed as part of the run-time system of HAWAII. The footprint monitor periodically saves the footprint to NVM. This approach helps to reduce the volume of footprints saved to NVM. In recovery, the footprint information is read from the NVM to the volatile main memory. The correct loop indices will be resumed based on the footprint information. Clearly, the system only needs to re-execute the last unfinished sub-operation after the resumption.

After the development of HAWAII, Kang et al. proposed an extension to the footprint-based approach called *model augmentation* [86]. The main idea is to append extra neural network components into the original DNN model to integrate progress indicators into the inference process. Thus, DNN outputs and progress indicators can be generated alternately, i.e., progress indicator preservation can be piggybacked onto DNN output preservation. The extra overhead of embedding progress information can be compensated by reducing NVM data backup overhead.

C. ADAPTIVE TECHNIQUES – ADAPTING ACCURACY WITH ENERGY AVAILABILITY

ePerceptive [87] is a framework for best-effort inference. The objective of ePerceptive is to adapt the computational complexity of DNN models to fluctuating energy conditions, a feature specific to inference in energy-harvesting scenarios. Several key techniques were combined to achieve the design goal. The ePerceptive framework is shown in Figure 5.

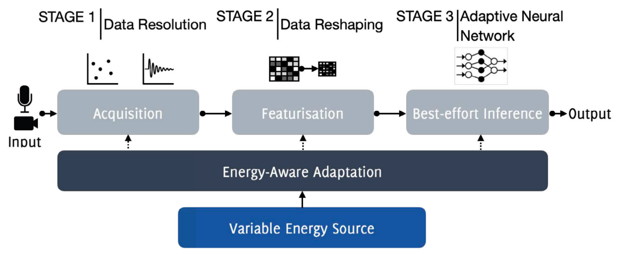


FIGURE 5. The framework of ePerceptive [87].

First, a new DNN model was developed to enable multi-resolution analysis. The basic idea is straightforward: if there is low energy, the resolution of the input data is reduced; thus, the energy consumption for DNN inference is reduced. This is a trade-off between analysis quality and energy availability. ePerceptive offers a single model that can operate at different input resolutions without storing multiple models in memory. The multi-resolution technique selects the input dimension and ensures the required accuracy with a specific energy budget.

Second, ePerceptive presented another technique which, at run time, selects the appropriate exit point of a DNN model for the instantaneous and unpredictable energy input to guarantee a timely response. ePerceptive enhances a MobileNetV1 DNN model with multiple exit points at intermediate positions throughout the network. At each exit point, a classifier is implemented to produce the output.

A research group from the University of Pittsburgh presented a series of techniques that adopt multi-exit neural networks for intermittent DNN inference [83], [84], [88]. As explained earlier, a DNN inference task must consider the available energy that is typically tiny and unstable. If the energy input is high, an inference task may finish before the energy is depleted; otherwise, the original inference task will not finish. For the second case, the inference task may exit earlier by skipping some of the latter layers in the original model. This ensures that the inference task will always finish with the available energy at run time, even if this may reduce analysis accuracy.

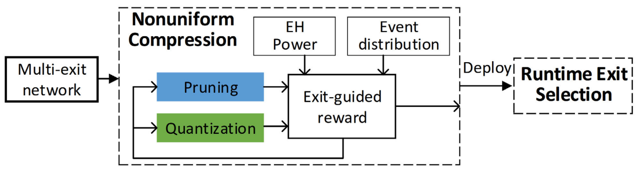


FIGURE 6. The framework of ePerceptive [83].

In [83], the authors start by compressing an existing large DNN (with layer-wise pruning and quantization) to fit into the limited computation capability of an MCU. However, most network compression algorithms only consider the accuracy of the final classifier. But for inference in the energy harvesting scenario, the accuracy of any exit point should be deemed comprehensively. This objective makes the proposed approach different from all other network compression techniques.

At run time, decisions must be made on where to exit according to the available energy input. There is an interesting trade-off in this decision. For example, when the energy input is low in the long run, even if the system has sufficient energy to exit at a later layer for the highest accuracy, a better decision can be to exit earlier and reserve some energy for the following inferences. In reality, the power trace and the inference distribution are unknown in advance. To select the best exit for each event, the authors proposed an online Q-learning approach to predict the information required for an early-exit decision.

The same research group implemented a multi-exit CNN modified from the above work. It deployed the new CNN on a small platform with an ARM Cortex-M0 processor core, 16KB SRAM as the main memory, and 128KB FLASH as NVM and is powered by a tiny solar panel [84], [88]. The experiments demonstrated that the multi-exit approach offers flexibility with a trade-off between accuracy and processing time, an essential feature desired for energy-harvesting devices with variable energy inputs and different operation situations.

D. NEURAL ARCHITECTURE SEARCH FOR MODELS TAILORED TO INTERMITTENT INFERENCE

Designing a high-quality DNN model is intractable regarding its design space. Traditionally, DNN models are manually designed with expert knowledge. However, this design approach is becoming increasingly unrealistic with the

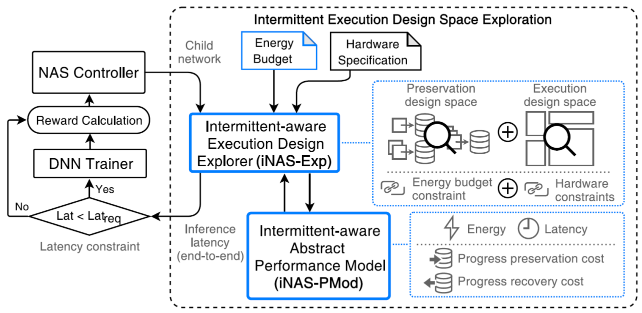

Intermittent Execution Design Space Exploration
Child network
Energy Budget
Hardware Specification
NAS Controller
Reward Calculation
DNN Trainer
Yes
No
Lat < Lat_req
Latency constraint
Intermittent-aware Execution Design Explorer (iNAS-Exp)
Preservation design space
Execution design space
Energy budget constraint
Hardware constraints
Inference latency (end-to-end)
Intermittent-aware Abstract Performance Model (iNAS-PMod)
Energy
Latency
Progress preservation cost
Progress recovery cost

FIGURE 7. The architecture of iNAS [97].

increasing complexity of the optimization objectives. Hence, neural architecture search (NAS) techniques [89] were developed to automatically search for highly accurate neural networks with specific design constraints.

A recent NAS approach called iNAS [97] was proposed to automatically find a DNN model with maximal accuracy and, most importantly, satisfying energy and latency constraints. Two key components realize the above design targets. The intermittent-aware Execution Design Explorer (iNAS-Exp) implements the design constraints on energy consumption and inference latency in the design space exploration step, which differentiates iNAS from other NAS approaches. iNAS-Exp ensures that for the resulting DNN model, 1) the maximal energy consumed by one layer should not exceed the available energy budget in each power cycle, and 2) the overall inference latency should not exceed a specified upper bound.

A design space explorer (DSE) cannot guarantee the quality of the resulting model. The critical cost parameters input to the DSE must be trustworthy. To solve this problem, iNAS presents an intermittent-aware Abstract Performance Model (iNAS-PMod). iNAS-PMod analytically formulates the power-cycle energy consumption and latency considering the additional costs, including the total cost of computing multiple tiles in a power cycle, the power-cycle energy consumption, the inference latency, and the recharging time. Specific to the intermittent execution scenario, the costs of rebooting, fetching progress indicators, and data re-fetching for progress recovery are modeled in the formulation of energy consumption and inference latency. Compared to the non-intermittent-aware NAS approach, iNAS generates DNN models that perform efficient and safe inference with energy and latency budgets.

E. DEPTH-FIRST EXECUTION FOR LOW CHECKPOINTING OVERHEAD

Generally, DNN inference is performed layer by layer, meaning each layer will generate a feature map and be used as input to the next layer. To ensure forward progress, the intermediate feature map data must be checkpointed to NVM to survive power failures. The checkpointing process presents a considerable energy and time overhead due to the size of feature map data.

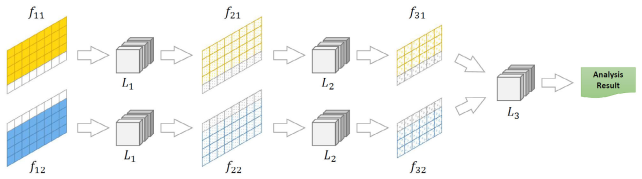


FIGURE 8. Depth-first DNN inference to optimize checkpointing overhead [90].

Lv and Xu proposed a depth-first intermittent inference approach to improve the checkpointing and execution efficiency [90]. Instead of executing the inference layer by layer, the input feature map is partitioned into small slices that span multiple layers, an example of which is shown in Figure 8. Since the energy required to execute a slice of the DNN is much smaller, the inference over one slice will probably finish before a power failure occurs. By this means, the intermediate feature maps need not be saved to NVM. This approach significantly reduces the overhead of checkpointing. For typical CNNs, the authors explored several dimensions to partition the network model, including array-wise partitioning, channel-wise partitioning, and kernel-wise partitioning. The maximal number of layers a slice can span is discussed, based on which the lower and upper bounds of the slice size can be determined. The depth-first inference may introduce extra overhead, as the rows (of the feature maps) at the slice boundary may be referenced twice by the two adjacent slices. However, their experiments showed that this extra overhead could be compensated by reducing the total data volume written to NVM.

F. TIMING MATTERS

Real-time requirements are a common feature of IoT applications, which is valid for those running on energy-harvesting devices. Classical real-time theories generally assume constant workload (the Worst-Case Execution Time for each task) and continuous execution on the CPU cores (the CPU cores are always on). However, both assumptions do not hold on energy harvesting devices. First, the energy generation of a harvester is bursty, not constant. Second, if the workload is a DNN inference task, the system may vary the analysis accuracy (for example, by the multi-exit approach) to adapt computation to the available energy. Thus, the workload also varies.

Islam et al. proposed Zygarde, a framework and a runtime system that enables time-aware execution of a DNN on intermittently-powered devices [98]. Zygarde contains five key components: a job generator, an energy manager, an agile DNN model, k-means classifiers, and a scheduler, shown in Figure 9. Sensor data are read into Zygarde and written to NVM, and then the data are analyzed by classification algorithms. The end-to-end processing of one data sample is called a job. The job generator creates jobs periodically and puts the jobs into a queue. The energy manager monitors the state of the energy storage as well as the energy harvesting

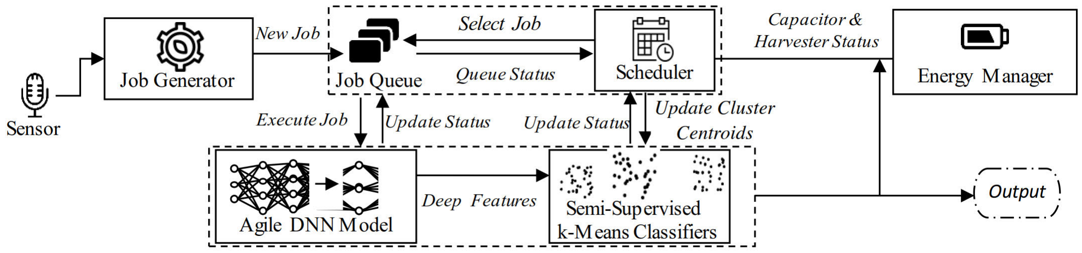


FIGURE 9. Zygarde system architecture [98].

rate that the scheduler will use for decision-making. The energy manager also contains a run time to enable jobs to progress across power failures. The agile DNN model is a DNN network, and rank decomposition and separation are applied to compress and fit the original network to the memory footprint. The model is essentially a multi-exit model, which means the model may exit earlier if the energy input is small. The k-means classifier takes the feature representation obtained by the DNN and generates the final classification result. Zygarde maintains a separate k-means classifier for each layer of the DNN. The most critical component is the scheduler. Zygarde implemented a dynamic priority real-time scheduler that considers the deadline and the expected accuracy. As the DNN may exit early, the scheduler partitions a job into mandatory and optional portions. The execution of the mandatory portion is to ensure both timeliness and accuracy. If extra energy and time remain, the optional portion will be executed to improve accuracy further.

G. HARDWARE DESIGN TO FACILITATE INTERMITTENT DNN INFERENCE

As the main overhead of intermittent DNN inference is to checkpoint program states to NVM, particular processors are developed to reduce the checkpointing overhead or even remove the need for checkpointing and improve the energy efficiency of DNN inference.

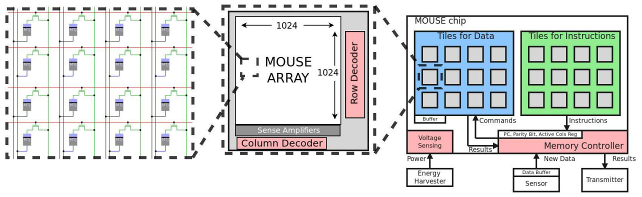
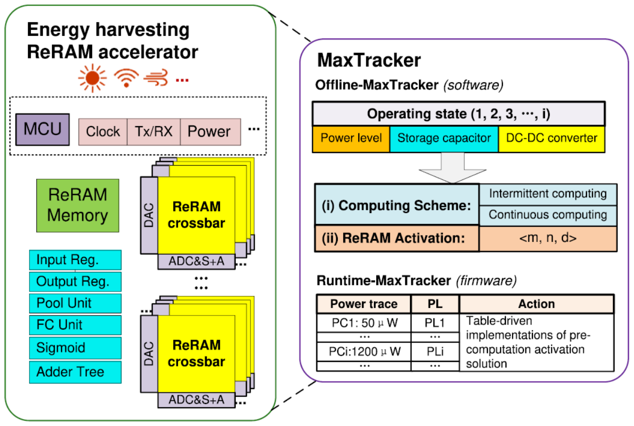


FIGURE 10. The hardware architecture of MOUSE [92].

MOUSE [92], [93] is an in-memory accelerator specifically designed to enable DNN inference in the presence of power failures, the architecture of which is shown in Figure 10. First, MOUSE has on-chip non-volatile memory to ensure data will not be lost across power failures, specifically the STT-MRAM array, which is considered a universal memory replacement [99]. MOUSE can be used as a standard STT-MRAM array and a computational substrate by minor modifications to the memory array. Besides the memory array, MOUSE has five other components to enable continuous computation in the presence of power failures: 1) a memory controller reading and issuing instructions, 2) an 128B memory buffer to facilitate reads/writes to the tiles, 3) a non-volatile program counter, 4) a non-volatile register as an instruction buffer, and 5) a voltage sensing circuitry to monitor the power source. Since MOUSE performs all computations within the non-volatile memory, program progress is saved after each operation. This makes restarting after the last instruction possible. To continue from the last instruction, MOUSE writes the program counter into a non-volatile register after each instruction. When restarts, MOUSE reads the next instruction from the address in the program counter. Program correctness in the presence of power failures has been proved regarding the hardware design. MOUSE has been demonstrated to be logically correct and energy-efficient in many applications, including those based on Binary Neural Network (BNN) models.



Power trace	PL	Action
PC1: 50 μW	PL1	Table-driven implementations of pre-computation activation solution
···	···	
PCi:1200 μW	PLi	
···	···	

FIGURE 11. The device and the framework of MaxTracker [94].

MaxTracker [94] is a software and hardware solution for CNN inference on energy-harvesting devices. The hardware of MaxTracker is a processor with specialized computing hardware components tailored for CNN inference and ReRAM memory to provide persistent data storage. The main idea behind MaxTracker is the so-called "Harvest-Store-Use" paradigm, by which a storage capacitor is conservatively filled with sufficient harvested energy to power the sensor node for a specified period of continuous operation before the compute phase. MaxTracker monitors the harvesting power levels and workload conditions at run time to decide the transition between computing modes. Under the intermittent computing mode, each power cycle can have multiple working cycles. A working cycle refers to a purely charging period followed by a consecutive working period. A working cycle is the unit of intermittent computing. Specifically, the status data will be backed up to the non-volatile ReRAM memory at the end of each working cycle. At the beginning of each working cycle, the stored status data will be read from the ReRAM to the volatile memory for progress. Experiments on CNN inference show that the MaxTracker approach can significantly enhance the program progress by precisely tracking the power status of the system.

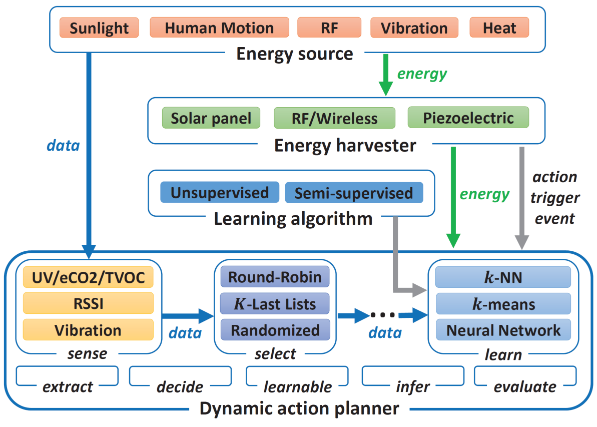


FIGURE 12. **The intermittent learning framework [95].**

H. LEARNING ON ENERGY-HARVESTING DEVICES

Most researchers believe an EH-IoT device with resource-constrained hardware and tiny energy input can only perform DNN inference. The resource-consuming training process should be performed in advance on computer servers. Some recent research practices explore the possibility of putting inference and training on an EH-IoT device. Lee et al. proposed the concept of "intermittent learning" that deploys both training and inference tasks on EH-IoT devices [95]. The objective is to enable the lifetime evolution of the DNN model and thus to improve the level of intelligence of the device.

As surveyed before, performing inference on intermittently powered EH-IoT devices is already challenging. Putting the more resource-consuming training onto the same device will bring many new challenges. First, as the energy input of an EH-IoT device is scarce, the tiny energy must be intelligently distributed between training and inference. Second, an intermittent learning system should smartly discard data samples at run time, as not all samples contribute equally to learning. Thus, unnecessary energy waste on training can be minimized. Third, greedily consuming the energy may cause the system to miss real-world events that deserve to learn when the system pauses. So the scheduling of the learning processes and the energy usage becomes essential. Although this article focuses on DNN inference on EH-IoT devices, we still include intermittent learning in this sub-section to extend our understanding.

Lee et al. proposed a framework to tackle the above-mentioned challenges [95], shown in Figure 12. Behind the framework, a programming model was proposed to develop an intermittent learning application. The programming model is called *action-based*, similar to the task-based approach. There are eight actions defined in the presented framework, including *sense*, *extract*, *decide*, *select*, *learnable*, *learn*, *evaluate*, *infer*. These actions are the major behaviors that an EH-IoT device may perform. A learning task comprises a subset of the actions that must be executed in a specific order.

The framework has a key component called the *dynamic action planner* that determines a sequence of actions at run time. When enough energy is harvested, the planner selects the best action that should be performed. Several performance indicators are defined to decide whether to learn or to infer, including the learning rate, the inference rate, and the combination of the two. The performance indicators are evaluated with the system's progress, and the objective can be tuned by adjusting the desirable indicator values at run time.

Learning example selection is another critical component of intermittent learning. The framework defines a series of mathematical criteria for selecting examples, including *uncertainty*, *balance*, *diversity*, and *representation*. Selecting the training examples to satisfy all the criteria is computationally expensive. The framework proposes several heuristics for different optimization purposes. To ensure *balance*, the selected examples fall into k clusters in a round-robin fashion. To ensure diversity and representation, two k-element lists are maintained that keep track of the last $2k$ examples that were selected and not selected, respectively. To ensure uncertainty, an example x_i with a probability of p_i.

Islam et al. studied the problem of training an efficient DNN model to run on intermittently powered EH-IoT devices [96]. Their work considered the various vector operations supported by the low-energy accelerators in the training process. By improving the efficiency of such accelerators, the DNN inference can make better progress with limited energy input. In their work, a resource-aware DNN training framework is proposed that comprises several components, including architecture search, model compression, normalization, and fixed point calculation. A software system is also designed that provides an efficient implementation flow for application developers. An on-demand checkpointing mechanism is proposed, which can minimize progress overhead in the presence of power failures.

I. A SHORT SUMMARY

Within the past several years, research on deploying deep learning on EH-IoT devices has touched many aspects and is forming a road map for the future. Early work makes it possible to correctly checkpoint DNN inference tasks [24], [85], but achieving low checkpointing cost remains unsatisfactory.

Although new chip hardware has been proposed to simplify checkpointing [92], [93], [94], excessive memory access due to checkpointing still stands for a major overhead. Related work considered adaptive DNN inference to fit the computation workload into the energy envelope (work surveyed in Section III-C). However, existing work only considered reducing inference computation with an early exit in the cases of low-energy input. More flexible adaptive DNN inference techniques are expected to compensate for the analysis quality if the energy becomes abundant. Due to the lack of precise energy prediction methods, the run-time decision on adaptive DNN inference remains unoptimized.

IV. FUTURE CHALLENGES AND NEW RESEARCH DIRECTIONS

Thanks to the research work listed in the previous section, intermittent DNN inference has been made possible on resource-constrained IoT devices. However, challenges remain in several aspects and need further exploration:

- Minimizing the overhead of checkpointing
- Dynamically adjusting the inference workload to maximize analysis accuracy with fluctuating energy input
- Efficiently and precisely predicting the energy harvesting power to guide adaptive inference

A. MINIMIZING THE OVERHEAD OF CHECKPOINTING

DNN inference tasks bring an unprecedented challenge to state checkpointing. Beyond register files and memory stack, DNN inference typically generates massive feature map data when it progresses layer by layer. For an inference task to continue execution after a power failure, the generated feature maps must be saved to NVM. This will incur significant energy overhead by writing a large amount of data to NVM. As a result, very little energy is left for the system to run useful work, i.e., DNN inference.

Re-scheduling the computation of an inference task deserves further investigation to reduce the amount of feature map data to be checkpointed. A first research practice in this line is the depth-first inference approach [90]. Note that feature maps are not the expected result of the inference but are used as intermediate data between layers. Once a layer has consumed the feature map, it can be safely discarded from the main memory. Although the technique in [90] works for CNNs, new challenges remain for DNNs with complex data dependency among layers, such as RNNs. New slicing approaches need to be explored.

Re-scheduling comes with extra overhead. By dividing the input data into small chunks and letting each chunk go through the convolution layers one by one, the filter data (i.e., weight parameters) will be reloaded and used each time a chunk goes through the layers. This will introduce extra energy overhead. Therefore, there is a need to balance reduced checkpointing of feature map data and increased reloading of filter data. The trade-off will be explored to find the design choice with minimal overhead.

Another design decision is to decide the size of a chunk. The number of times to reload weight parameters during an inference will be very large to partition the input data into small chunks. If the chunk is too big, the probability of encountering power failures during the execution of a chunk is increased. So, chunk size design should adapt to the availability of the energy that can be harvested soon. Implementation techniques that allow changing chunk size at run time need to be explored.

B. FLEXIBLE RUN-TIME ADAPTIVE DNN INFERENCE

One significant characteristic of energy harvesting is that the harvesting power may vary in an extensive range due to the changing ambient environment. A DNN inference may take a long time to finish in cases where energy harvesting is very low. This is unacceptable in applications where the analysis results are expected to be produced within a given deadline. For such cases, system designers may sacrifice analysis precision by executing only part of the DNN to reduce the time and energy consumed to finish an inference.

Even though the amount of energy input can be predicted, it is still possible that the predicted amount of energy is lower than that is harvested. In this case, a pessimistic adaptation will be decided at the beginning of an inference to aggressively reduce the inference workload by executing a tiny portion of the DNN. At the end of the inference, there is remained energy. Such energy should have been used to execute a less pruned network for better precision. Therefore, we need techniques that, when extra energy is available, can perform additional computation on the pruned part of the DNN to compensate for the analysis precision. This line of work will improve the existing multi-exit approaches.

For early-exit networks, if it turns out that the inference exits too early, we can let the inference go further along the network layers. This requires the output of the last convolution layer (take CNN, for example) to be saved so the inference can continue from where it has exited. Note that keeping the output may increase the amount of data to be checkpointed. Then a balance between the overhead of extra checkpointing and the actual precision gain obtained by the compensation should be explored.

For filter pruning techniques, it is very hard to do compensation once the inference is finished. This is because, at the end of each layer, the outputs of all filters at this layer are merged into a single output of the current layer. If we want to compensate by adding the pruned filters at this layer, then all the computation after this layer should be redone. One opportunity to solve this problem will be to do filter pruning structurally. For example, one can perform less filter pruning at earlier layers of the DNN to avoid losing information too early. When the inference goes deeper in the network, more aggressive filter pruning can be considered to reduce computation. A more aggressive idea would be to significantly change the typical CNN architecture into a multi-path architecture so that each path may contribute partly to the analysis precision, and the contribution of each path can be added up.

Another potential line of adaptive inference is on the frame level. If after an inference is done, there is still extra energy, one can let the IoT device immediately do additional sensing and use the extra energy to perform inference on the extra sensor data. In some application scenarios, multiple less precise analysis results may work the same as few more accurate results. This approach can be used at least for some applications to achieve higher overall precision instead of maximizing the accuracy of a single inference.

Recently, collaborative DNN inference is becoming a hot topic in the IoT domain [100]. The main idea is to offload some workload of DNN inference from the IoT device to the edge server so that the energy consumption can be reduced with less inference work to do. This DNN inference paradigm may help alleviate the intermittent power supply issue on EH-IoT devices and thus increase the probability of a successful inference. However, collaborative DNN inference requires to transfer intermittent data from the edge server to the IoT device, which may incur extra energy consumption for wireless communication. It would be interesting to investigate which decision may lead to the optimal result: to perform the whole DNN inference task on the IoT device or to pay the price of communication for less computation overhead.

C. EFFICIENT RUN-TIME ENERGY PREDICTION

Predicting the amount of energy that can be harvested in the near future is critical to decision-making. For depth-first execution (to reduce state checkpoint overhead), the amount of available energy will affect the decision on the granularity of a chunk; for adaptive run-time inference (to trade precision for low energy consumption), the availability of energy will affect the decision of reduced execution, e.g., how early to exit, and thus affect the analysis precision that can be achieved. Therefore, run-time energy prediction is an indispensable capability.

Existing work on energy prediction stands at two extremes of complexity. Complex approaches leverage learning approaches, including DNNs, to predict future incoming energy by analyzing large volume history data on energy harvesting [101]. Such approaches are suitable for long-term or offline energy prediction (such as predicting energy input for large solar arrays for a considerable time scale). Still, they are not a good option for runtime prediction on low-energy EH-IoT devices because the computation involved in the prediction will bring a significant overhead to the system. In other related research practices of intermittent computing, very simple approaches are adopted which predict future energy harvesting by looking at the harvesting power of the previous power cycle [102]. Although the computation overhead of such methods is very small, the prediction precision will be severely compromised. It needs to investigate new energy prediction approaches that are efficient enough to perform at run time and as precise as possible.

The key to efficient and precise energy prediction is to combine history/experience information and run-time information. To this end, pattern-based energy prediction can be investigated. The rationale behind this idea is that the energy harvesting power for a short while does not change arbitrarily in reality; instead, patterns exist [103]. A pattern specifies how the energy harvesting power changes over a short period. One may seek to model the energy harvesting patterns abstractly with mathematical formulations explicitly. At design time, parameterized patterns of different energy harvesting sources (solar, wind, heat, etc.) will be explored and modeled with the help of machine learning over a large amount of historical data. Each pattern, specified by some mathematical formulation, will be characterized by a small set of critical indicators. At run time, the values of the key indicators will be extracted from the energy harvesting information recorded for the past few power cycles, and the energy harvesting pattern will be identified. The values of the key indicators are also fed into the parameterized pattern formulation to predict the amount of energy that can be harvested for a given period in the future. Thus, the EH-IoT device does not need to record a long energy harvesting history or undergo complex analysis to predict energy input.

V. CONCLUSION

Future IoT devices will become more intelligent (using DNNs to analyze sensor data) and greener (eliminating traditional batteries and relying on energy harvesters). A design gap was created between computation and energy-intensive deep learning tasks and low-energy EH-IoT devices. To conduct deep learning on EH-IoT devices is an exciting research direction. This article surveys current research work in this domain. However, many issues still exist and need further exploration. So, this article also calls for research to crack the enabling techniques to bridge the gap, specifically, enabling deep learning tasks to intermittently execute on EH-IoT devices and make reliable progress, enabling deep learning to dynamically balance between analysis precision and computation overhead to cope with changing energy input. These capabilities are pillared by efficient and precise run-time energy input prediction. We believe the success in this research domain will unlock many new IoT applications in the future, making the IoT more intelligent and environmentally friendly.

REFERENCES

[1] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, computing, and communications for energy harvesting IoTs: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1222–1250, 2020, doi: 10.1109/COMST.2019.2962526.

[2] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent computing: Challenges and opportunities," in *Proc. 2nd Summit Adv. Program. Lang. (SNAPL)*, B. S. Lerner, R. Bodík, and S. Krishnamurthi, Eds., Asilomar, CA, USA, vol. 71, 2017, pp. 8:1–8:14, doi: 10.4230/LIPIcs.SNAPL.2017.8.

[3] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," in *Proc. 16th ASPLOS*, 2011, pp. 159–170.

[4] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," in *Proc. 36th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2015, vol. 50, no. 6, pp. 575–585.

[5] J. van der Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," in *Proc. 12th OSDI*, 2016, pp. 17–32.

[6] H. Jayakumar, A. Raha, and V. Raghunathan, "QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers," in *Proc. 27th Int. Conf. VLSI Design 13th Int. Conf. Embedded Syst.*, Jan. 2014, pp. 330–335.

[7] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Hibernus++: A self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 12, pp. 1968–1980, Jun. 2016.

[8] M. Hicks, "Clank: Architectural support for intermittent computation," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, vol. 45, no. 2, pp. 228–240.

[9] N. A. Bhatti and L. Mottola, "HarvOS: Efficient code instrumentation for transiently-powered embedded sensing," in *Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2017, pp. 209–220.

[10] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *Proc. 13th OSDI*, 2018, pp. 129–144.

[11] K. Maeng and B. Lucia, "Supporting peripherals in intermittent systems with just-in-time checkpoints," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2019, pp. 1101–1116.

[12] J. Choi, H. Joe, Y. Kim, and C. Jung, "Achieving stagnation-free intermittent computation with boundary-free adaptive execution," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2019, pp. 331–344.

[13] V. Kortbeek, K. S. Yildirim, A. Bakar, J. Sorber, J. Hester, and P. Pawełczak, "Time-sensitive intermittent computing meets legacy software," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 85–99.

[14] A. Colin and B. Lucia, "Chain: Tasks and channels for reliable intermittent programs," in *Proc. ACM SIGPLAN Int. Conf. Object-Oriented Program., Syst., Lang., Appl.*, Oct. 2016, pp. 514–530.

[15] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," in *Proc. OOPSLA*, 2017, pp. 1–30.

[16] K. S. Yıldırım, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawełczak, and J. Hester, "InK: Reactive kernel for tiny batteryless sensors," in *Proc. 16th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2018, pp. 41–53.

[17] J. Hester, K. Storer, and J. Sorber, "Timely execution on intermittently powered batteryless sensors," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2017, pp. 1–13.

[18] E. Ruppel and B. Lucia, "Transactional concurrency control for intermittent, energy-harvesting computing systems," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2019, pp. 1085–1100.

[19] S. Liu, W. Zhang, M. Lv, Q. Chen, and N. Guan, "LATICS: A low-overhead adaptive task-based intermittent computing system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3711–3723, Nov. 2020, doi: 10.1109/TCAD.2020.3012214.

[20] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 92:1–92:36, Sep. 2018, doi: 10.1145/3234150.

[21] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, May 2022, doi: 10.1016/j.neucom.2021.04.141.

[22] J. Iannacci, "The WEAF mnecosystem: A perspective of MEMS/NEMS technologies as pillars of future 6G, super-IoT and tactile internet," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2021, pp. 52–59.

[23] J. Iannacci and H. V. Poor, "Review and perspectives of micro/nano technologies as key-enablers of 6G," *IEEE Access*, vol. 10, pp. 55428–55458, 2022.

[24] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 199–213, doi: 10.1145/3297858.3304011.

[25] S. Umesh and S. Mittal, "A survey of techniques for intermittent computing," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101859, doi: 10.1016/j.sysarc.2020.101859.

[26] J. Hester and J. Sorber, "The future of sensing is batteryless, intermittent, and awesome," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Delft, The Netherlands, Nov. 2017, p. 21, doi: 10.1145/3131672.3131699.

[27] A. Haeberlin, Y. Rösch, M. V. Tholl, Y. Gugler, J. Okle, P. P. Heinisch, T. Reichlin, J. Burger, and A. Zurbuchen, "Intracardiac turbines suitable for catheter-based implantation—An approach to power battery and leadless cardiac pacemakers?" *IEEE Trans. Biomed. Eng.*, vol. 67, no. 4, pp. 1159–1166, Apr. 2020, doi: 10.1109/TBME.2019.2932028.

[28] S. Chalasani and J. M. Conrad, "A survey of energy harvesting sources for embedded systems," in *Proc. IEEE SoutheastCon*, Apr. 2008, pp. 442–447.

[29] V. Kartsch, S. Benatti, M. Mancini, M. Magno, and L. Benini, "Smart wearable wristband for EMG based gesture recognition powered by solar energy harvester," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[30] Z. Yun and S. Han, "Towards a real-time wireless powered communication network: Design, implementation and evaluation," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2021, pp. 347–359, doi: 10.1109/RTAS52030.2021.00035.

[31] (2022). *Prometheus: Energy Harvesting Made Easy*. [Online]. Available: https://www.matrixindustries.com/technology

[32] N. Shah, L. Kamdar, D. Gokalgandhi, and N. Mehendale, "Walking pattern analysis using deep learning for energy harvesting smart shoes with IoT," *Neural Comput. Appl.*, vol. 33, no. 18, pp. 11617–11625, Sep. 2021, doi: 10.1007/s00521-021-05864-4.

[33] V. Devabhaktuni, M. Alam, S. S. S. R. Depuru, R. C. Green, D. Nims, and C. Near, "Solar energy: Trends and enabling technologies," *Renew. Sustain. Energy Rev.*, vol. 19, pp. 555–564, Mar. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032112006363

[34] (2022). *The Matrix Industries Homepage*. [Online]. Available: https://www.matrixindustries.com/

[35] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 757–789, 2nd Quart., 2015, doi: 10.1109/COMST.2014.2368999.

[36] S. Kim, R. Vyas, J. Bito, K. Niotaki, A. Collado, A. Georgiadis, and M. M. Tentzeris, "Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms," *Proc. IEEE*, vol. 102, no. 11, pp. 1649–1666, Nov. 2014.

[37] M. Aldrigo, M. Dragoman, M. Modreanu, I. Povey, S. Iordanescu, D. Vasilache, A. Dinescu, M. Shanawani, and D. Masotti, "Harvesting electromagnetic energy in the V-band using a rectenna formed by a bow tie integrated with a 6-nm-thick Au/HfO_2/Pt metal–insulator–metal diode," *IEEE Trans. Electron Devices*, vol. 65, no. 7, pp. 2973–2980, May 2018.

[38] N. Xing and G. A. Rincon-Mora, "Highest wireless power: Inductively coupled or RF?" in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2020, pp. 298–301.

[39] X. Li, C.-Y. Tsui, and W.-H. Ki, "A 13.56 MHz wireless power transfer system with reconfigurable resonant regulating rectifier and wireless power control for implantable medical devices," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 978–989, Apr. 2015.

[40] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 11, pp. 2608–2615, Nov. 2008, doi: 10.1109/TIM.2008.925019.

[41] M. Gorlatova, P. R. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman, "Energy harvesting active networked tags (EnHANTs) for ubiquitous object networking," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 18–25, Dec. 2010, doi: 10.1109/MWC.2010.5675774.

[42] H. Vocca and F. Cottone, "Kinetic energy harvesting," in *ICT-Energy-Concepts Towards Zero*, G. Fagas, L. Gammaitoni, D. Paul, and G. A. Berini, Eds. Rijeka, Croatia: IntechOpen, 2014, ch. 3, doi: 10.5772/57091.

[43] Q. Huang, Y. Mei, W. Wang, and Q. Zhang, "Toward battery-free wearable devices: The synergy between two feet," *ACM Trans. Cyber Phys. Syst.*, vol. 2, no. 3, pp. 20:1–20:18, 2018, doi: 10.1145/3185503.

[44] M. Thielen, L. Sigrist, M. Magno, C. Hierold, and L. Benini, "Human body heat for powering wearable devices: From thermal energy to application," *Energy Convers. Manage.*, vol. 131, pp. 44–54, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890416310007

[45] C. Huang, T. Tan, Z. Wang, S. Zhang, F. Yang, Z. Lin, and Z. Yan, "Origami dynamics based soft piezoelectric energy harvester for machine learning assisted self-powered gait biometric identification," *Energy Convers. Manage.*, vol. 263, Jul. 2022, Art. no. 115720. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890422005167

[46] T. Tan, Z. Yan, H. Zou, K. Ma, F. Liu, L. Zhao, Z. Peng, and W. Zhang, "Renewable energy harvesting and absorbing via multi-scale metamaterial systems for Internet of Things," *Appl. Energy*, vol. 254, Nov. 2019, Art. no. 113717. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261919314047

[47] M. Badr, M. M. Aboudina, F. A. Hussien, and A. N. Mohieldin, "Simultaneous multi-source integrated energy harvesting system for IoE applications," in *Proc. IEEE 62nd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, H. Lee and R. L. Geiger, Eds., Dallas, TX, USA, Aug. 2019, pp. 271–274, doi: 10.1109/MWSCAS.2019.8884893.

[48] X. Cui, J. Zhang, H. Zhou, and C. Deng, "PowerPool: Multi-source ambient energy harvesting," in *Proc. 6th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Jul. 2020, pp. 86–90.

[49] W. Zhou, X. Wang, C. Hu, Q. Li, C. Li, L. Du, and H. Yu, "Research on multi-source environmental micro energy harvesting and utilization," in *Proc. 6th Asia Conf. Power Electr. Eng. (ACPEE)*, Apr. 2021, pp. 1072–1076.

[50] (2022). *The Kinergizer Homepage*. [Online]. Available: https://kinergizer.com/

[51] K. Vijayaraghavan and R. Rajamani, "Novel batteryless wireless sensor for traffic-flow measurement," *IEEE Trans. Veh. Technol.*, vol. 59, no. 7, pp. 3249–3260, Sep. 2010, doi: 10.1109/TVT.2010.2050013.

[52] F. Li, Y. Yang, Z. Chi, L. Zhao, Y. Yang, and J. Luo, "Trinity: Enabling self-sustaining WSNs indoors with energy-free sensing and networking," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 2, pp. 57:1–57:27, 2018, doi: 10.1145/3173039.

[53] B. Lu, Y. Chen, D. Ou, H. Chen, L. Diao, W. Zhang, J. Zheng, W. Ma, L. Sun, and X. Feng, "Ultra-flexible piezoelectric devices integrated with heart to harvest the biomechanical energy," *Sci. Rep.*, vol. 5, no. 1, p. 16065, Dec. 2015.

[54] (2022). *The Solepower Homepage*. [Online]. Available: http://www.solepowertech.com/

[55] (2022). *The Instepnanopower Homepage*. [Online]. Available: http://www.instepnanopower.com/

[56] (2022). *The Bionic Power Homepage*. [Online]. Available: https://www.bionic-power.com/

[57] *Semiconductor Devices—Semiconductor Devices for Energy Harvesting and Generation—Part 1: Vibration Based Piezoelectric Energy harvesting*, document IEC 62830-1:2017, 2022. [Online]. Available: https://webstore.iec.ch/publication/27039

[58] P. K, U. Govindarajan, V. K. Ramachandaramurthy, and B. Jeevarathinam, "Integrating solar photovoltaic energy conversion systems into industrial and commercial electrical energy utilization—A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 39–54, Jun. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X17300742

[59] C. Schuss, B. Eichberger, and T. Rahkonen, "Impact of solar radiation on the output power of moving photovoltaic (PV) installations," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I MTC)*, May 2018, pp. 1–6.

[60] P. Singla and S. R. Sarangi, "A survey and experimental analysis of checkpointing techniques for energy harvesting devices," *J. Syst. Archit.*, vol. 126, May 2022, Art. no. 102464, doi: 10.1016/j.sysarc.2022.102464.

[61] *MSP430 Microcontrollers*. Accessed: Oct. 15, 2022. [Online]. Available: http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html

[62] M. Surbatovich, L. Jia, and B. Lucia, "I/O dependent idempotence bugs in intermittent systems," in *Proc. OOPSLA*, 2019, pp. 1–31.

[63] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[64] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., 2014, pp. 1724–1734, doi: 10.3115/v1/d14-1179.

[65] S. Branco, A. G. Ferreira, and J. Cabral, "Machine learning in resource-scarce embedded systems, FPGAs, and end-devices: A survey," *Electronics*, vol. 8, no. 11, p. 1289, Nov. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/11/1289

[66] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," *CoRR*, vol. abs/1602.07360, Feb. 2016.

[67] 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," 2017, *arXiv:1704.04861*.

[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[69] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 1314–1324, doi: 10.1109/ICCV.2019.00140.

[70] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 6848–6856.

[71] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient CNN architecture design," in *Proc. 15th Eur. Conf.*, vol. 11218, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Munich, Germany. Cham, Switzerland: Springer, Sep. 2018, pp. 122–138, doi: 10.1007/978-3-030-01264-9_8.

[72] Z. Li and L. Meng, "A survey of model pruning for deep neural network," in *Proc. 4th Int. Symp. Adv. Technol. Appl. Internet Things (ATAIT)*, H. Nishikawa and X. Kong, Eds., vol. 3198, Aug. 2022, pp. 25–34. [Online]. Available: http://ceur-ws.org/Vol-3198/paper4.pdf

[73] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021, doi: 10.1016/j.neucom.2021.07.045.

[74] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021, doi: 10.1007/s11263-021-01453-z.

[75] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "MCDNN: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., Services*, R. K. Balan, A. Misra, S. Agarwal, and C. Mascolo, Eds., Singapore, Jun. 2016, pp. 123–136, doi: 10.1145/2906388.2906396.

[76] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, and V. Kostakos, Eds., Los Cabos, Mexico, Oct. 2019, p. 54, doi: 10.1145/3300061.3345455.

[77] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., Long Beach, CA, USA, Dec. 2017, pp. 2181–2191.

[78] X. Gao, Y. Zhao, L. Dudziak, R. D. Mullins, and C. Xu, "Dynamic channel pruning: Feature boosting and suppression," *CoRR*, vol. abs/1810.05331, Jun. 2018.

[79] J. Yu, L. Yang, N. Xu, J. Yang, and T. S. Huang, "Slimmable neural networks," *CoRR*, vol. abs/1812.08928, Jul. 2018.

[80] W. Hua, Y. Zhou, C. D. Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., Vancouver, BC, Canada, Dec. 2019, pp. 1884–1894.

[81] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *Proc. 34th Int. Conf. Mach. Learn.*, D. Precup and Y. W. Teh, Eds., Sydney, NSW, Australia, vol. 70, Aug. 2017, pp. 527–536.

[82] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Cancún, Mexico, Dec. 2016, pp. 2464–2469, doi: 10.1109/ICPR.2016.7900006.

[83] Y. Wu, Z. Wang, Z. Jia, Y. Shi, and J. Hu, "Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218526.

[84] Y. Li, Y. Wu, X. Zhang, J. Hu, and I. Lee, "Energy-aware adaptive multi-exit neural network inference implementation for a millimeter-scale sensing system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 7, pp. 849–859, Jul. 2022, doi: 10.1109/TVLSI.2022.3171308.

[85] C.-K. Kang, H. R. Mendis, C.-H. Lin, M.-S. Chen, and P.-C. Hsiu, "Everything leaves footprints: Hardware accelerated intermittent deep inference," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3479–3491, Nov. 2020, doi: 10.1109/TCAD.2020.3012217.

[86] C.-K. Kang, H. R. Mendis, C.-H. Lin, M.-S. Chen, and P.-C. Hsiu, "More is less: Model augmentation for intermittent deep inference," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 5, pp. 1–26, Sep. 2022, doi: 10.1145/3506732.

[87] A. Montanari, M. Sharma, D. Jenkus, M. Alloulah, L. Qendro, and F. Kawsar, "EPerceptive: Energy reactive embedded intelligence for batteryless sensors," in *Proc. 18th Conf. Embedded Netw. Sensor Syst.*, Nov. 2020, pp. 382–394, doi: 10.1145/3384419.3430782.

[88] Y. Li, Y. Wu, X. Zhang, E. Hamed, J. Hu, and I. Lee, "Developing a miniature energy-harvesting-powered edge device with multi-exit neural network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Daegu, South Korea, May 2021, pp. 1–5, doi: 10.1109/ISCAS51556.2021.9401799.

[89] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017. [Online]. Available: https://openreview.net/forum?id=r1Ue8Hcxg

[90] M. Lv and E. Xu, "Efficient DNN execution on intermittently-powered IoT devices with depth-first inference," *IEEE Access*, vol. 10, pp. 101999–102008, 2022, doi: 10.1109/ACCESS.2022.3203719.

[91] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 4, no. 3, pp. 1–29, Sep. 2020, doi: 10.1145/3411808.

[92] S. Resch, S. K. Khatamifard, Z. I. Chowdhury, M. Zabihi, Z. Zhao, H. Cilasun, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, "MOUSE: Inference in non-volatile memory for energy harvesting applications," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Athens, Greece, Oct. 2020, pp. 400–414, doi: 10.1109/MICRO50266.2020.00042.

[93] S. Resch, S. K. Khatamifard, Z. I. Chowdhury, M. Zabihi, Z. Zhao, H. Cilasun, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, "Energy efficient and reliable inference in nonvolatile memory under extreme operating conditions," *ACM Trans. Embedded Comput. Syst.*, Mar. 2022, doi: 10.1145/3520130.

[94] K. Qiu, N. Jao, K. Zhou, Y. Liu, J. Sampson, M. T. Kandemir, and V. Narayanan, "MaxTracker: Continuously tracking the maximum computation progress for energy harvesting reram-based CNN accelerators," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, pp. 78:1–78:23, 2021, doi: 10.1145/3477009.

[95] S. Lee, B. Islam, Y. Luo, and S. Nirjon, "Intermittent learning: On-device machine learning on intermittently powered system," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 4, pp. 141:1–141:30, 2019, doi: 10.1145/3369837.

[96] S. Islam, J. Deng, S. Zhou, C. Pan, C. Ding, and M. Xie, "Enabling fast deep learning on tiny energy-harvesting IoT devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 921–926, doi: 10.23919/DATE54114.2022.9774756.

[97] H. R. Mendis, C. Kang, and P. Hsiu, "Intermittent-aware neural architecture search," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, pp. 64:1–64:27, 2021, doi: 10.1145/3476995.

[98] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 3, pp. 82:1–82:29, 2020, doi: 10.1145/3411808.

[99] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," in *Proc. 45th Annu. Conf. Design Autom.*, Anaheim, CA, USA, 2008, pp. 554–559, doi: 10.1145/1391469.1391610.

[100] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[101] M. Chu, H. Li, X. Liao, and S. Cui, "Reinforcement learning-based multiaccess control and battery prediction with energy harvesting in IoT systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2009–2020, Apr. 2018, doi: 10.1109/JIOT.2018.2872440.

[102] A. Y. Majid, C. D. Donne, K. Maeng, A. Colin, K. S. Yildirim, B. Lucia, and P. Pawełczak, "Dynamic task-based intermittent execution for energy-harvesting devices," *ACM Trans. Sensor Netw.*, vol. 16, no. 1, pp. 1–24, Feb. 2020, doi: 10.1145/3360285.

[103] A. Bakar and J. Hester, "The energy harvesting mode abstraction," in *Proc. 16th ACM Conf. Embedded Networked Sensor Syst.*, G. S. Ramachandran and B. Krishnamachari, Eds., Shenzhen, China, Nov. 2018, pp. 418–419.

MINGSONG LV (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, China, in 2002, 2005, and 2010, respectively. He is currently an Associate Professor with Northeastern University. His research interests include timing analysis of real-time systems, intermittent computing, and satellite edge computing. He received the Best Paper Award of DATE 2013.

ENYU XU received the B.S. degree in computer science from the Taiyuan University of Science and Technology, China, in 2019, and the M.S. degree in computer science from Northeastern University, China, in 2022. His research interests include intermittent computing, deep learning, and timing analysis of DNN programs on GPU.