Energy-Aware Integrated Neural Architecture Search and Partitioning for Distributed Internet of Things (IoT)

Baichuan Huang, *Student Member, IEEE*, Azra Abtahi, *Member, IEEE*, and Amir Aminifar, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) are one of the key enablers of personalized health. However, IoT devices often have stringent constraints in terms of resources, e.g., energy budget, and, therefore, limited possibilities to exploit the state-of-the-art Deep Neural Networks (DNNs). Energy-aware Neural Architecture Search (NAS) is proposed to tackle this challenge, by exploring lightweight DNN (DNN) architectures on a single IoT device, but not leveraging the inherently distributed nature of IoT systems. As a result, the joint optimization of DNN architectures and DNN computation partitioning/offloading has not been addressed to date. In this paper, we propose an energy-aware NAS framework for distributed IoT, aiming to search for distributed Deep Neural Networks (DNNs) to maximize prediction performance subjected to Flash Memory (Flash), Random Access Memory (RAM), and energy constraints. Our framework searches for lightweight DNN architecture with optimized prediction performance and its corresponding optimal computation partitioning to offload the partial DNN from edge to fog in a joint optimization. We evaluate our framework in the context of two common health applications, namely, seizure detection and arrhythmia classification, and demonstrate the effectiveness of our proposed joint optimization framework compared to NAS benchmarks.**

***Index Terms*—Energy-aware neural architecture search (NAS), computation offloading, mobile edge computing, distributed computing, battery-powered Internet of Things (IoT), low-power IoT, low-power wearables, and energy optimization.**

I. Introduction

TODAY, Internet of Things (IoT) devices are one of the key enablers of real-time and long-term health monitoring on a personalized basis [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. IoT devices, however, often have stringent energy budgets and constrained resources to exploit modern Deep Neural Networks (DNNs), which limits their adoption in real-world applications.

Neural Architecture Search (NAS) has been leveraged to automatically design lightweight Deep Neural Networks (DNNs) for IoT devices [14], [15], [16], [17]. In particular, resource-aware NAS has been exploited to search DNN architectures, taking into account the requirements in terms of Flash Memory (Flash), Random Access Memory (RAM), and energy on various hardware devices [18], [19], [20]. However, current resource-aware NAS only pays attention to individual hardware devices such as Microcontroller (MCU) boards [21], [22], [23], [24], [25], [26], mobile devices [16], [17], [27], server Central Processing Units (CPUs) and Graphics Processing Cards (GPUs) [16], [17], [20].

The classical deployment of DNNs searched by NAS is on individual IoT devices, hence not exploiting the inherently distributed nature of IoT systems. However, DNNs can be partitioned into multiple parts and offloaded to edge, fog, and cloud in order to minimize energy consumption [28], [29], [30], [31], [32], [33], [34]. The optimal partition point for DNN computation partitioning is decided based on the DNN architectures, the IoT devices, and the wireless communication [28]. Previous studies on DNN computation partitioning [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44] are mainly limited to representative DNNs such as VGG [45], AlexNet [46], and MobileNet [47]. These representative DNNs are, however, not designed/optimized for distributed IoT systems with stringent energy and memory constraints, leading to suboptimal solutions.

While both resource-aware NAS and DNN computation partitioning has been considered individually in the prior work, the joint optimization of DNN architectures and DNN computation partitioning/offloading has not been addressed to date. Considering either resource-aware NAS or DNN computation partitioning often leads to suboptimal solutions. Moreover, considering both resource-aware NAS and DNN computation partitioning, but not in an integrated/co-design fashion, e.g., first performing resource-aware NAS and after DNN computation partitioning, will also lead to suboptimal solutions. This is because the decisions regarding the DNN architecture are fixed without considering the computation partitioning alternative,

Received 2 July 2024; revised 2 October 2024; accepted 28 October 2024. Date of publication 7 November 2024; date of current version 26 November 2024. This work was supported in part by the Swedish Wallenberg AI, Autonomous Systems and Software Program (WASP) and in part by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at C3SE through the Swedish Research Council (VR) under Grant 2022-06725. The review of this article was arranged by Associate Editor Pi-Cheng Hsiu. *(Corresponding author: Baichuan Huang.)*

The authors are with the Department of Electrical and Information Technology, Lund University, 22363 Lund, Sweden (e-mail: baichuan.huang@eit.lth.se).

Digital Object Identifier 10.1109/TCASAI.2024.3493036

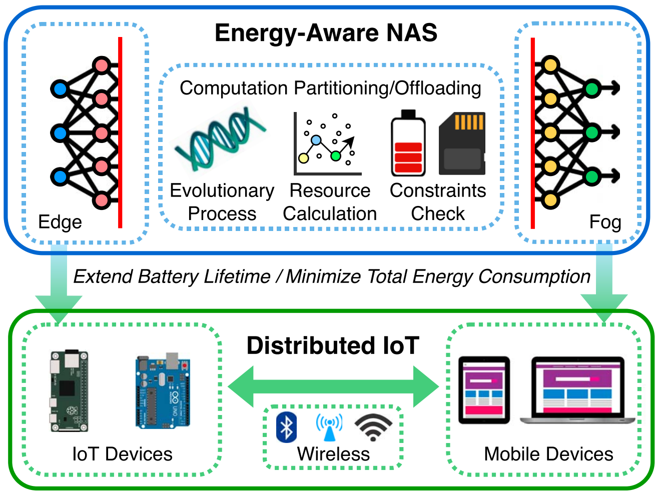


Fig. 1. Overview of our proposed energy-aware NAS framework for distributed IoT, with joint optimization of DNN architectures and DNN computation partitioning/offloading.

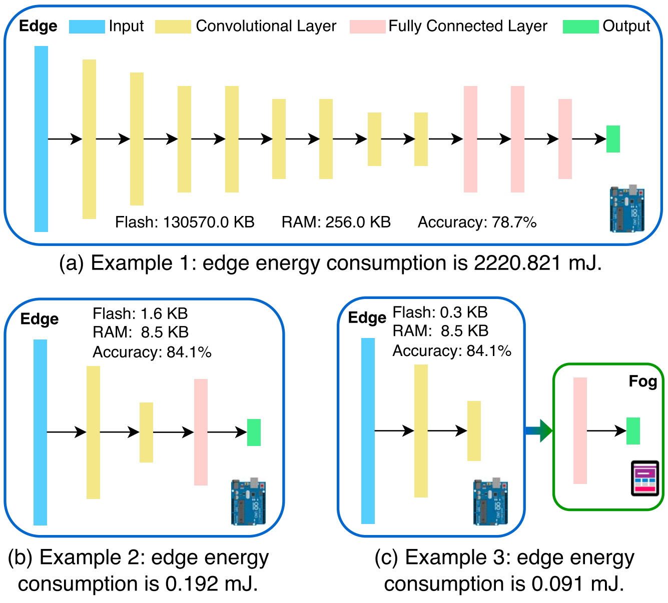


Fig. 2. Motivational examples to show the demand for joint optimization of DNN architectures and DNN computation partitioning/offloading.

which is a greedy (as in "greedy algorithm") strategy often leading to suboptimal solutions.

We take three examples, as shown in Fig. 2, to further motivate the need for the joint optimization of DNN architectures and DNN computation partitioning/offloading. These three examples are in the context of epileptic seizure detection applications [48].

- *Example 1 – Standard VGG11:* Let us consider VGG11, one of the main representative architectures in the DNNs domain, shown in Fig. 2(a). The edge energy consumption of VGG11 running on the edge node is 2220.821 Millijoule (mJ). The Flash footprint is 130570.0 Kilobyte (KB) and the RAM footprint is 256.0 KB, which exceeds the capacity of typical IoT devices such as STM32L476RG (ARM Cotex-M4) and other microcontrollers based on ARM Cortex-M family. Even if the representative VGG11 is partitioned, the total energy consumption still does not meet the energy budget of typical IoT devices.
- *Example 2 – Energy-Aware NAS:* Let us now consider energy-aware NAS to identify lightweight DNN architectures, considering one IoT device. The entire DNN architecture identified by NAS is deployed on the edge node, as shown in Fig. 2(b). When compared to the representative VGG11, the Flash footprint of Example 2 is 1.6 KB (reduced by 81606.2 times), and the RAM footprint of Example 2 is 8.5 KB (reduced by 30.1 times). Moreover, the edge energy consumption is reduced by 11566.7 times from 2220.821 mJ to 0.192 mJ.
- *Example 3 – Joint Optimization of DNN Architectures and DNN Computation Partitioning/Offloading:* Let us next consider the same DNN architecture in Example 2, but partitioned/offloaded on distributed IoT infrastructure. Our proposed framework partitions the searched DNN and offloads the partial DNN from the edge node to the fog node. As shown in Fig. 2(c), the Flash footprint of Example 3 is 0.3 KB, which is reduced by 5.3 times compared to Example 2. Furthermore, the edge energy consumption of Example 3 is 0.091 mJ, which is reduced by 2.1 times compared to Example 2. This is, essentially, because the energy overheads of processing the final two layers on the edge node are more than the energy overheads of transmitting the output of the third layer between the edge and fog nodes.

We conclude that the standard DNNs architectures consume large amounts of energy and memory, beyond the constraints of typical IoT devices. At the same time, considering only energy-aware NAS or computation partitioning leads to suboptimal solutions. Therefore, it is essential and possible to jointly optimize DNN architectures and DNN computation partitioning/offloading to reduce the energy-overheads of DNNs on distributed IoT infrastructure.

In this article, for the first time to the best of our knowledge, we propose an energy-aware NAS for distributed IoT, aiming to search for DNN architecture and its corresponding optimal computation partitioning to maximize prediction performance subjected to the constrained resources (i.e., Flash, limited RAM, and energy), as shown in Fig. 1. Our proposed framework jointly optimizes (1) the DNN architectures and (2) the DNN partitioning/offloading on the distributed IoT infrastructure in an integrated fashion, to avoid suboptimal solutions. We evaluate our framework in the context of two common health applications, namely, seizure detection and arrhythmia classification, and demonstrate the effectiveness of our proposed joint optimization framework.

Our main contributions are summarized below:

- We propose an energy-aware NAS for distributed IoT, which aims to search for DNN architecture and its corresponding optimal computation partition point to maximize prediction performance subject to the constrained resources. To ensure efficient search space exploration, we explore evolutionary frameworks (Genetic Algorithms) and hardware-aware energy consumption models and speed up the optimization process up to 32 times by exploiting the parallelism provided by multi-core CPUs and GPUs platforms. The proposed joint optimization is not limited to Genetic Algorithms and can be seamlessly incorporated into various NAS methods [17], [49], [50], [51], [52], [53], [54].

- We evaluate our energy-aware NAS framework based on two real-world medical IoT applications with wearable technologies, namely, seizure detection [48] and arrhythmia classification [55]. The accuracy of seizure detection reaches 87.3% with the edge energy consumption of 0.375 mJ. The accuracy of arrhythmia classification reaches 88.0% with the edge energy consumption of 0.270 mJ. Furthermore, when compared to the representative VGG11 [45], 8225.3 times the edge energy consumption is saved for arrhythmia classification, and 5058.8 times the total energy consumption is saved for seizure detection using our proposed framework. Finally, we compare our proposed NAS framework to NAS benchmarks including NASNet [49] (Reinforcement Learning), DARTS [50] (Gradient-Based), SPOS [17] (One-Shot), CNN-GA [53] (Evolutionary Algorithms), and training-free NAS (Zero-Shot NAS [54]), on seizure detection and arrhythmia classification. The results demonstrate that our proposed NAS framework achieves a significantly more lightweight DNN architecture with comparable accuracy and, in some cases, even better accuracy.

The remainder of this paper is organized as follows. In Section II, we review the literature on NAS, resource-aware NAS, and DNN computation partitioning. Next, in Section III, we propose our energy-aware NAS for distributed IoT in detail. Then, in Section IV, we describe the experimental setup including datasets, edge platform, and implementation details. Next, in Section V, we experimentally evaluate our energy-aware NAS and compare our proposed framework against several state-of-the-art NAS benchmarks. Finally, Section VI serves as the conclusion of this work.

II. Related Work

In this section, we briefly review the literature on NAS, resource-aware NAS, and DNN computation partitioning.

A. NAS

NAS aims to automatically design DNN architectures rather than directly exploiting representative DNN architectures [14]. NAS techniques are typically Reinforcement Learning (RL)-based NAS, Gradient-based NAS, and Evolutionary Algorithm (EA)-based NAS. RL-based NAS methods require large amounts of GPUs and consume major energy [15]. Gradient-based NAS methods require building up a supernet first. The supernet needs to be designed manually by experts [16], [17]. EA-based NAS methods, on the other hand, have attracted a lot of attention due to their capability to explore a large number of potential architectures and the acceleration by parallel evaluation [56]. However, classical NAS does not consider the memory constraints and energy budget of IoT devices.

B. Resource-Aware NAS

Resource-Aware NAS focuses on searching DNNs, while taking into consideration the resource constraints of IoT devices [18], [19], [57]. In [20], a transformable architecture search method is proposed for IoT devices. Similarly, there are several studies on NAS for individual IoT devices [21], [22], [23], [24], [25], [26], [27]. In [21], MCUNet aims to design an efficient DNN on the tiny MCU by TinyNAS and TinyEngine. In [22], a toolkit for DNN inference on MCU is proposed. In [23], MCUNetV2 finds the imbalanced memory distribution of Convolutional Neural Network (CNN) and proposes the patch-based inference for the memory-intensive stage of CNN on MCU. In [24], MicroNets targets searching the Deep Neural Networks (DNNs) for commodity MCU, but considering the number of operations as a proxy for latency. In [25], SpArse combines NAS with pruning for MCU. In [26], μNAS incorporates model compression and pruning for MCU. Furthermore, in [58] and [59], NSGA-Net and NSGANetV2 consider multiple objectives including the error metric and computational complexity. In [60], MOEA-PS exploits the multi-objective evolutionary algorithm considering the two objectives of precision and execution time. In [61], [62], the NAS under resource constraints approach can search for optimal network models without wasting the limited resources available. In relation to the energy-aware NAS, in [63], the joint optimization is over the neural architecture and quantization space, considering multiple objectives, i.e., minimizing both the classification error and the energy consumption. In [64], ChamNet searches the neural architectures considering the prediction accuracy and latency/energy by efficient predictive models. In [65], MONAS jointly optimizes the prediction accuracy and energy consumption by reinforcement learning.

However, these current resource-aware NAS algorithms only focus on individual hardware devices such as the specified MCU boards [21], [22], [23], [24], [25], [26], mobile devices [16], [17], [27], or server CPUs and GPUs [16], [17], [20]. Therefore, the state of the art in resource-aware NAS does not exploit the inherently distributed nature of IoT systems.

C. DNN Computation Partitioning

DNN computation partitioning techniques offload DNNs over distributed IoT infrastructure, from edge, to fog, and perhaps even cloud, to satisfy the constraints of Flash, RAM and allocated energy budget. In [28], a layer-level computation partitioning strategy is firstly proposed to reduce latency and save energy consumption for devices with CPUs and GPUs. In [37], a distributed inference framework for the CPUs and GPUs simultaneously saves energy and reduces latency on mobile devices. In [8], a self-aware classification technique is introduced to minimize energy overheads of classical machine learning techniques by offloading complex computation to fog and/or cloud engines. In [38], an adaptive deep learning approach is applied to reduce the inference latency by dividing the computation of the entire DNN into distributed IoT systems. Overall, partitioning/offloading is incorporated into many mobile edge computing systems [40], [41], [42], [43], [44]. However, current studies [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44] only consider representative and fixed DNNs, and do not optimize the DNN architectures.

Therefore, joint optimization of DNN architectures and DNN computation partitioning/offloading has not been considered to date. Our paper proposes an energy-aware NAS for distributed IoT, jointly optimizing DNN architectures and DNN computation partitioning/offloading for the first time, to the best of our knowledge.

III. Energy-Aware NAS for Distributed IoT

In this section, we describe our proposed energy-aware NAS for distributed IoT in detail. We consider two scenarios, the edge energy scenario and the total energy scenario, and formulate the optimization problem subject to the constrained resources for each of these scenarios in Section III-A. Then, we propose the overall flow of our framework in Section III-B.

A. Problem Formulation

IoT devices have a limited energy budget and constrained resources for long-term health monitoring. Modern representative DNNs become deeper and deeper with higher energy consumption and memory footprint. These DNNs do not perfectly match the constrained memory and energy budget of IoT devices. Although resource-aware NAS is leveraged to automatically design heterogeneous DNNs, it does not make use of the inherently distributed nature of IoT systems. Furthermore, current DNN computation partitioning techniques only use representative DNNs, which still consume large amounts of energy. In this paper, we jointly optimize the DNN architectures and DNN computation partitioning/offloading, formulated as follows:

$$\begin{aligned} \max_{A,P} \quad & \mathbb{P}(A,P) \\ s.t. \quad & \mathbb{E}(A,P) \le \mathbb{E}_{limit} \\ & \mathbb{R}(A,P) \le \mathbb{R}_{limit} \\ & \mathbb{F}(A,P) \le \mathbb{F}_{limit}, \end{aligned} \tag{1}$$

where the goal is to maximize the prediction performance $\mathbb{P}$. The inputs of this joint optimization problem are the limits on energy consumption $\mathbb{E}_{limit}$, Flash footprint $\mathbb{F}_{limit}$, RAM footprint $\mathbb{R}_{limit}$ for the specific health monitoring application and IoT device. The outputs of the joint optimization problem are the searched architecture A and the partition point P (P divides A into two parts to be deployed on edge and fog nodes). The actual energy consumption $\mathbb{E}(A, P)$, the actual RAM footprint $\mathbb{R}(A, P)$, and the actual Flash footprint $\mathbb{F}(A, P)$ for the searched architecture A and the partition point P are estimated, and they are required to be bounded by $\mathbb{E}_{limit}$, $\mathbb{R}_{limit}$, and $\mathbb{F}_{limit}$, respectively.

We consider two different scenarios for formulating our optimization problem. The first one is the edge energy scenario where we consider the battery lifetime of edge nodes to be constrained and the second one is the total energy scenario where we consider the total energy consumption of distributed IoT systems to be constrained.

In the edge energy scenario, the energy consumption of DNN inference on edge nodes and on wireless communication, i.e., $\mathbb{E}_{edge}$, are taken into consideration. The partition point P divides the DNN into two parts: one partial DNN is from layer 0 to layer P ($N_{0:P}$), and another is from layer P+1 to the end layer ($N_{P+1:end}$). The energy consumption of the partial DNN $N_{0:P}$ on edge nodes is denoted as $\mathbb{E}_{edge}^{comp}$. As for wireless communication, the energy consumption $\mathbb{E}_{edge}^{comm}$ depends on the length of layer P, i.e., l_P. The energy consumption on edge nodes is denoted by $\mathbb{E}_{edge}$, which incorporates $\mathbb{E}_{edge}^{comp}$ and $\mathbb{E}_{edge}^{comm}$, and determines its battery lifetime. We aim to search for DNNs with the maximum prediction performance subjected to the constraints of Flash, RAM, and allocated edge energy budget. Hence, the optimization problem in the edge energy scenario can be formulated as follows:

$$\begin{aligned} \max_{A,P} \quad & \mathbb{P}(A,P) \\ s.t. \quad & \mathbb{E}_{edge}(A,P) \le \mathbb{E}_{limit} \\ & \mathbb{R}(A,P) \le \mathbb{R}_{limit} \\ & \mathbb{F}(A,P) \le \mathbb{F}_{limit}, \end{aligned} \tag{2}$$

where $\mathbb{E}_{edge}(A,P) = \mathbb{E}_{edge}^{comp}(N_{0:P}) + \mathbb{E}_{edge}^{comm}(l_P)$. Note that, even in this scenario, we leverage the inherently distributed nature of IoT infrastructure. This is because $\mathbb{E}_{edge}(A,P) = \mathbb{E}_{edge}^{comp}(N_{0:P}) + \mathbb{E}_{edge}^{comm}(l_P) \le \mathbb{E}_{edge}^{comp}(N_{0:end})$ or alternatively $\mathbb{E}_{edge}^{comm}(l_P) \le \mathbb{E}_{edge}^{comp}(N_{P+1:end})$. That is, the energy required for the communication of partial results to the fog is less than running the partial network on the edge.

In the total energy scenario, the energy consumption of DNN inference on edge nodes and fog nodes and wireless communication are taken into consideration. While the partial DNN $N_{0:P}$ runs on edge nodes, the subsequent partial DNN $N_{P+1:end}$ from layer $P+1$ to the end layer runs on fog nodes. The energy consumption on fog nodes is denoted as $\mathbb{E}_{fog}$. We aim to search for DNNs with the maximum prediction performance subjected to the constraints of Flash, RAM, and allocated total energy budget. Thus, the optimization problem in the total energy scenario can be formulated as follows:

$$\begin{aligned} \max_{A,P} \quad & \mathbb{P}(A,P) \\ s.t. \quad & \mathbb{E}_{total}(A,P) \le \mathbb{E}_{limit} \\ & \mathbb{R}(A,P) \le \mathbb{R}_{limit} \\ & \mathbb{F}(A,P) \le \mathbb{F}_{limit}, \end{aligned} \tag{3}$$

where $\mathbb{E}_{total}(A,P) = \mathbb{E}_{edge}(A,P) + \mathbb{E}_{fog}(A,P)$. Similar to the edge energy scenario, we have $\mathbb{E}_{edge}(A,P) = \mathbb{E}_{edge}^{comp}(N_{0:P}) + \mathbb{E}_{edge}^{comm}(l_P)$. In addition, we have $\mathbb{E}_{fog}(A,P) = \mathbb{E}_{fog}^{comp}(N_{P+1:end}) + \mathbb{E}_{fog}^{comm}(l_P)$. Because of the inherent heterogeneity of the IoT infrastructure, the fog node may be significantly more efficient, in terms of energy

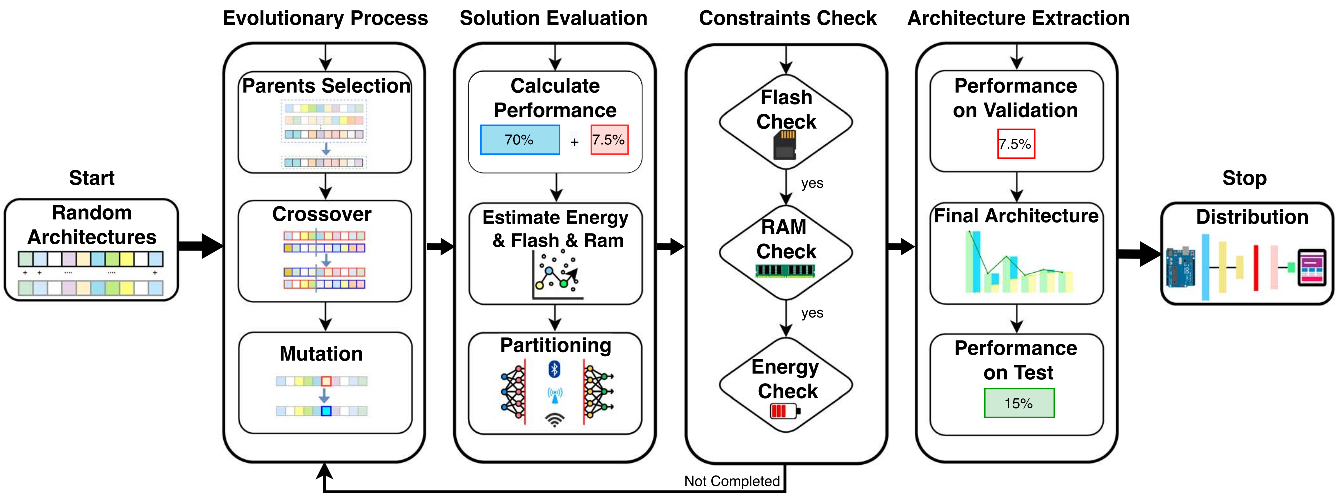


Fig. 3. Overall flow of our proposed energy-aware NAS for distributed IoT comprises four main steps. The first three steps are repeated for a certain iteration. The final architecture is extracted in the last step.

consumption, in the inference process than the edge node. That is, $\mathbb{E}^{comp}_{edge}(N_{P+1:end}) \gg \mathbb{E}^{comp}_{fog}(N_{P+1:end})$. This, in turn, may result in $\mathbb{E}^{comp}_{edge}(N_{0:P}) + \mathbb{E}^{comm}_{edge}(l_P) + \mathbb{E}^{comp}_{fog}(N_{P+1:end}) + \mathbb{E}^{comm}_{fog}(l_P) \leq \mathbb{E}^{comp}_{edge}(N_{0:end})$. Therefore, leveraging the inherent heterogeneity of the IoT infrastructure can result in major energy savings.

In this paper, our primary focus is on CNN-based models when addressing DNNs; however, the proposed approach can be easily extended to Fully Connected Neural Networks [66]. For instance, Fully Connected Neural Networks are a special type of CNN where the filter size equals the input size.

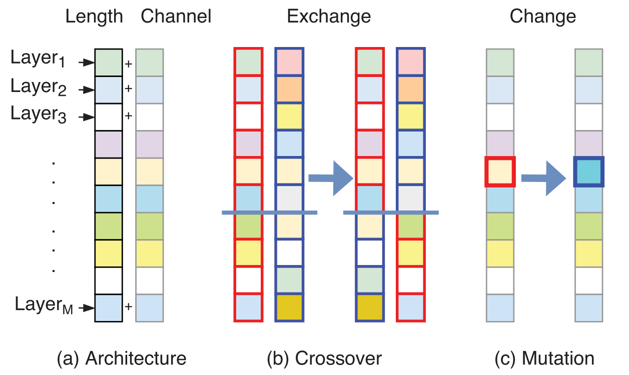


Fig. 4. Evolutionary process with the ability of layer optimization. Each architecture has two chromosomes: *Length* chromosome and *Channel* chromosome. The number of neurons in each layer is denoted as *Length* and the channel number of each layer is denoted as *Channel*.

B. The Overall Flow of Our Proposed Energy-Aware NAS

The overall flow of our proposed energy-aware NAS is illustrated in Fig. 3. To solve the proposed optimization problem in each of the mentioned scenarios, the overall flow of our proposed framework includes four steps: evolutionary process, solution evaluation, constraints check, and architecture extraction. The first three steps are performed iteratively until a desired solution is identified. Then, the final architecture is extracted in the last step.

1) Evolutionary Process: This step is to search the design space of the proposed problem to identify the DNN architectures. Since the design space exploration process is inherently a combinatorial problem, here we resort to evolutionary algorithms to search the design space. Initially, the architectures are generated randomly. For the next iterations, the inputs are the architectures of the previous generation. The outputs are the newly evolutionary architectures. Our proposed energy-aware NAS is based on EA [67] and Genetic Algorithm (GA) [68]. More specifically, each DNN architecture has two chromosomes. One chromosome is for the length of each layer, and another chromosome is for the number of channels of each layer, as shown in Fig. 4(a). Different colors represent different genes in the corresponding search space, and the white color means 0, i.e., skipping this layer. The maximum possible number (M) of DNN layers is fixed, but with the ability of layer optimization, the final number of DNN layers may be less.

For evolving new architectures in the next generation, parent selection with the tournament strategy is exploited for breeding. We conduct several tournaments to breed several parents. In each tournament, only the architecture with the highest prediction performance wins and this architecture is selected as one of the parents. The corresponding architectures are regarded as the winners/parents and passed to Crossover and Mutation. In Crossover as shown in Fig. 4(b), the chromosomes from two different parents exchange the partial chromosome below the Crossover line, which is denoted in dark blue. In Mutation as shown in Fig. 4(c), the gene circled in red is mutated to another gene randomly in the search space for the channel. The Crossover and Mutation are done separately for both the length chromosome and the channel chromosome. After Crossover and Mutation, a new generation of architectures is obtained.

2) Solution Evaluation: This step is to evaluate each design solution by estimating the prediction performance, Flash footprint, RAM footprint, and energy consumption. The inputs are the evolutionary architectures and the output is the actual resource consumption considering the partitioning point.

For each searched architecture, the prediction performance needs to be estimated. To this end, the dataset is divided into three parts: train data, validation data, and test data, to avoid potential data snooping and overfitting. To estimate the prediction performance of each searched architecture, we, first, train a model for each searched architecture. Then, we evaluate this model on the validation set to obtain the estimated prediction performance. Moreover, in the estimation of the prediction performance, half of the validation data (randomly selected as a *Bagging* method) and train data are exploited to choose the best DNN in the training process, and the remaining half of validation data are utilized to evaluate the performance of generalization in *Architecture Extraction*. This mechanism, known as *Bagging*, is used to reduce the chances of overfitting to the validation data during the evolutionary process.

Next, for each searched architecture, the required amount of resources, in terms of Flashs, RAM, and energy need to be estimated. The static DNN weights stored in the Flash of edge nodes are used to calculate the Flash footprint. For calculating the RAM footprint, the activation buffer, i.e., the maximum memory footprint of neurons between every two successive layers is used [24]. On the other hand, although real-time measurement of energy consumption on hardware devices is accurate, the real-time measurement in NAS process consumes large amounts of time and expensive labor [69]. To ensure efficient design/search space exploration, a prediction model for hardware-aware energy consumption at the granularity of a single layer is proposed. Hardware devices such as MCU and mobile devices sequentially execute each DNN layer. Thus, the prediction model for hardware-aware energy consumption at layer granularity is reliable to predict the energy consumption for the entire DNN on such hardware devices [70].

To efficiently estimate the energy overhead $\mathbb{E}_{edge}^{comp}$ of each searched architecture during the evolutionary process, in the offline phase, the prediction model for hardware-aware latency is built by sampling various DNN layers from the search space. More specifically, to obtain the hardware-aware latency at layer granularity, CNN and Fully Connected Network (FC) are considered. The actual latency has a nonlinear relationship with the Floating Point Operations (FLOPs) in the search space. For predicting the layer-wise latency of CNN and FC, surrogate hardware-aware energy consumption models are exploited. For latency prediction of CNN $\mathbb{L}_{CNN}$, the searched length l_{in} and the searched channel c_{in} of the input layer, and the searched length l_{out} and the searched channel c_{out} of the output layer are required. For latency prediction of FC $\mathbb{L}_{FC}$, only l_{in} and l_{out} are needed. Additionally, the latency of Transposed CNN (TCNN) can be formulated based on $\mathbb{L}_{CNN}$ because the TCNN is regarded as a variant of CNN. Next, using the least-squares method, the surrogate power consumption is estimated. Finally, the power of DNN inference on edge nodes is measured and the estimated energy consumption $\mathbb{E}_{edge}^{comp}$ is the product of latency and the corresponding power.

In addition to $\mathbb{E}_{edge}^{comp}$, to estimate the edge energy overheads of each searched architecture, we also need to estimate the energy of wireless communication between the edge and the fog, captured by $\mathbb{E}_{edge}^{comm}$. This value, however, depends on the partitioning point P and the output size of the layer at which the DNN is partitioned/offloaded. Our proposed DNN computation partitioning/offloading takes the searched DNN apart and offloads the partial DNN from edge to fog according to the optimal partition point for each input architecture. Once the optimal partitioning point is identified, the value of the energy overheads of wireless communication $\mathbb{E}_{edge}^{comm}$ may be estimated. This strategy aims to minimize the energy consumption on edge energy consumption on the distributed IoT systems. This sub-optimization problem to minimize $\mathbb{E}_{edge}(A,P) = \mathbb{E}_{edge}^{comp}(N_{0:P}) + \mathbb{E}_{edge}^{comm}(l_P)$ in the edge energy scenario is formulated as follows:

$$\begin{aligned}\min_{P} \quad & \mathbb{E}_{edge}(A,P)\\ s.t. \quad & \mathbb{R}(A,P) \leq \mathbb{R}_{limit}\\ & \mathbb{F}(A,P) \leq \mathbb{F}_{limit},\end{aligned} \tag{4}$$

where the optimal value of partitioning point P can be obtained using an efficient exhaustive search.

In the total energy scenario, the total energy overheads are given by $\mathbb{E}_{total}(A,P) = \mathbb{E}_{edge}(A,P) + \mathbb{E}_{fog}(A,P)$. Therefore, in addition to the edge energy $\mathbb{E}_{edge}(A,P)$, we also need to estimate the $\mathbb{E}_{fog}^{comp}(N_{P+1:end})$ in a similar fashion as in $\mathbb{E}_{edge}^{comp}(N_{0:P})$. When the fog node is significantly more efficient than the edge node in terms of energy consumption, partitioning/offloading the DNN architecture from edge to fog minimizes the total energy overheads, $\mathbb{E}_{total}(A,P) = \mathbb{E}_{edge}(A,P) + \mathbb{E}_{fog}(A,P) = \mathbb{E}_{edge}^{comp}(N_{0:P}) + \mathbb{E}_{edge}^{comm}(l_P) + \mathbb{E}_{fog}^{comp}(N_{P+1:end}) + \mathbb{E}_{fog}^{comm}(l_P)$, leveraging the inherently distributed nature of IoT systems. However, as captured by $\mathbb{E}_{total}(A,P)$ above, this also depends on the energy overheads of communication. In essence, the optimal partitioning point has to be selected carefully in such a way that the energy gains of offloading the partial DNN to the fog is more than the energy overheads of communication between the edge and the fog. The optimal value of partitioning point P can be obtained using an efficient exhaustive search as follows,

$$\begin{aligned}\min_{P} \quad & \mathbb{E}_{total}(A,P)\\ s.t. \quad & \mathbb{R}(A,P) \leq \mathbb{R}_{limit}\\ & \mathbb{F}(A,P) \leq \mathbb{F}_{limit}.\end{aligned} \tag{5}$$

3) Constraints Check: In this step, for each searched architecture and partitioning/offloading point, we investigate whether the constraints w.r.t. Flash, RAM, and energy are satisfied. The inputs are the searched architectures, their partitioning/offloading point, and their corresponding resources in terms of Flash, RAM, and energy. The outputs are the architectures, which are not eliminated, and their corresponding adjusted prediction performance. We envision three scenarios:

1) Valid Solutions: If the solution (searched architecture and its corresponding partitioning/offloading point) satisfies all constraints, we consider the solution with its original

fitness K, given by its prediction performance, without any adjustment.

2) Abandoned Solutions: If the solution does not satisfy the constraints of Flash or RAM, the corresponding architecture is abandoned.
3) Invalid Solutions: If the solution satisfies the constraints of Flash and RAM, but does not satisfy the energy budget, the fitness value of the solution needs to be adjusted, as discussed below.

To guide the evolutionary process towards more energy-efficient solutions, here we propose an adjustment procedure for the solution that satisfies the constraints of Flash and RAM, but does not satisfy the energy budget. In essence, we penalize these solutions to have a worse fitness than all valid solutions, but we do not abandon them to maintain the diversity in the population. Furthermore, the higher the energy consumption of an invalid solution, the lower the adjusted fitness value $\mathbb{P}^{\circ}_{adjusted}$ should be. This is to reduce the probability of being selected in the parent selection process. The adjusted fitness (prediction performance) is given by: $\mathbb{P}^{\circ}_{adjusted} = \mathbb{P}^{\top}_{min} - \frac{\mathbb{P}^{\top}_{min} - \mathbb{P}^{\top}_{bound}}{\mathbb{E}^{\circ}_{max} - \mathbb{E}_{limit}} \cdot (\mathbb{E}^{\circ} - \mathbb{E}_{limit})$, where $\mathbb{E}^{\circ}$ is the energy overheads of the invalid solution and $\mathbb{P}^{\top}_{bound}$ is the theoretical minimum prediction performance for specified applications. $\mathbb{E}^{\circ}_{max}$ and $\mathbb{P}^{\top}_{min}$ denote the maximum energy consumption among all invalid solutions and the minimum prediction performance of all valid solutions, respectively.

The first three steps are performed iteratively until a desired solution is identified. Then, the best architecture identified in the evolutionary process is extracted, as discussed in the following.

4) Architecture Extraction: This step is to extract the best solution to be deployed on the distributed IoT infrastructure. The inputs are the valid solutions, namely, architectures and partitioning points, identified in the *Evolutionary Process* and *Solution Evaluation*, respectively. The output is the best solution satisfying the constraints of Flash, RAM, and the allocated energy budget. First, the remaining half of the validation data are utilized to perform and choose the best solution with the highest prediction performance that satisfies the constraints. Once the best solution is identified, the final average prediction performance for this solution is obtained considering the unseen test data. This solution will be adopted to be deployed on the distributed IoT system.

IV. Experimental Setup

A. Datasets

To evaluate our energy-aware NAS framework, we consider two real-world medical IoT applications with wearable technologies, namely, seizure detection based on the CHB-MIT Scalp Electroencephalogram (EEG) Dataset [48] and arrhythmia classification based on the MIT-BIH Arrhythmia Dataset [55]. Both datasets in our experiments are balanced, and accuracy, i.e., the total number of correctly classified inputs divided by the total number of inputs is used as the metric of prediction performance. The datasets are divided into train data (70%), validation data (15%), and test data (15%).

1) CHB-MIT Scalp EEG Dataset: This dataset contains 23 cases from 22 patients (5 males and 17 females) with epilepsy. Only two channels (T7F7 and T8F8) are considered, to be consistent with the wearable IoT devices for real-time seizure monitoring [71], [72], [73], [74]. We do not consider patients 6, 14, and 16 because they have very short-lasting seizures. A bandpass filter with a pass-band of 1–30 Hz is applied to the raw EEG signals. The filtered signal is segmented with a window length of 4 seconds, i.e., 1024 samples with a Z-score standardization [75]. Finally, the EEG signals of two channels are concatenated into 2048 dimensions, as the input of DNNs.

2) MIT-BIH Arrhythmia Dataset: This dataset encompasses Electrocardiogram (ECG) samples from 47 different patients with cardiovascular problems. In accordance with the AAMI EC57 standard [76], five different types of arrhythmias are categorized by beat annotations. The raw data are pre-processed by windowing, normalizing, finding local maximums, finding ECG R-peak, and finding R-R time intervals [77]. To ensure the same input length, we upsample each arrhythmia sample to the length of 2048. Finally, the dataset used in our experiments is balanced by randomly taking the same number of different arrhythmia samples among five types.

TABLE I
Search Space and Performance of Our Prediction Model of Hardware-Aware Energy Consumption

Network	Input	Search Space	R-squared
CNN	l_{in}/l_{out} c_{in}/c_{out}	[2048,1024,512,256,128,64,32,16,8,4] [4,3,2,1]	0.99
FC	l_{in}/l_{out}	[2048,1024,512,256,128,64,32,16,8,4]	0.99

B. Edge Platform

Our energy-aware NAS framework is tailored to resource-constrained IoT devices such as e-Glass [71] for seizure detection and the SmartCardia INYU wearable sensor [78] for arrhythmia classification. For the edge nodes, the MCU called STM32L476RG with ARM Cotex-M4 is exploited. This MCU board has 1 Megabyte (MB) of Flash and 128 KB of RAM and features as milliwatt-range power consumption. STM32CUbe.AI is used to convert the written codes to C codes for DNNs and implement DNNs on edge nodes for actual measurement of latency. Furthermore, the Otii Arc power analyzer is exploited to measure the hardware-aware power of edge nodes.

As discussed in Section III-B2, to be efficient in search space exploration, we consider a surrogate hardware-aware energy consumption model. For the search space in our surrogate model, as shown in Table I, the searchable length for input layer l_{in} and output layer l_{out} each has 10 different values. Especially for CNN, the searchable number of channels for input layer c_{in} and output layer c_{out} each has 4 different values. We sampled layers from the search space of all possible solutions and measured the actual latency of these layers. By using the least squares method, the surrogate hardware-aware energy consumption model is regressed. To evaluate the relevance of our surrogate model, we consider the Coefficient of Determination (R-squared) metric. The R-squared for both CNN and

FC reaches 0.99, which shows that our surrogate hardware-aware energy consumption model is reliable. However, our proposed framework is not restricted to this surrogate model and, essentially, any power model may be adopted to estimate the energy overheads of the design solutions search within our proposed framework.

Finally, Bluetooth Low Energy (BLE) is the wireless communication available on both e-Glass [71] and INYU wearable sensor [78] for transmission because of its low energy property. The power value of BLE is extracted from the Nordic DevZone power estimator [79].

C. Implementation Details

Our energy-aware NAS framework is implemented based on PyGAD (an open-source Python library for building GA) [80] and Ray (an open-source unified compute framework for parallelism) [81]. We implement the energy-aware NAS framework on the server of 2 x 16-core Intel(R) Xeon(R) Gold 6226R (Skylake) CPUs and 4 NVIDIA Tesla T4 GPUs, which allows a 32-fold acceleration with the aid of parallelism.

In addition, the length of each layer in the search space is $[1024, 512, 256, 128, 64, 32, 16, 8, 4, 0]$ and the channel number of each layer in the search space is $[4, 3, 2, 1]$. For training the DNNs based on TensorFlow Keras [82], the Adam optimizer is adopted. Moreover, for the application of seizure detection, we set the batch size to 4 and epochs to 20. For the application of arrhythmia classification, we set the batch size to 32 and epochs to 20. To avoid overfitting, dropout is exploited only in the training process and it does not change the latency and thus, has no influence on the surrogate hardware-aware energy consumption model.

V. Experimental Results

In this section, we discuss the experimental results of our proposed energy-aware NAS. First, the proposed energy-aware NAS for the edge energy scenario is presented in Section V-A. Then, the proposed energy-aware NAS for the total energy scenario is presented in Section V-B. After that, the ablation study is conducted to evaluate the performance of the layer optimization, the DNN computation partitioning/offloading, and the representative DNNs compared with the DNNs searched by our energy-aware NAS in Section V-C. Next, the comparison between our proposed NAS framework and NAS benchmarks is presented in Section V-D. Finally, the evolutionary process during the training phase is demonstrated in Section V-E.

A. Edge Energy Scenario

1) Seizure Detection: Here, we investigate the performance of the proposed energy-aware NAS in the edge energy scenario for seizure detection considering wearable systems. Table II shows the prediction performance (accuracy), edge energy, Flash, and RAM for four solutions. Each solution is obtained by one energy-aware NAS subjected to its edge energy consumption limit. As the edge energy consumption $\mathbb{E}_{edge}$ increases, the accuracy increase gradually. The edge energy consumption of 0.375 mJ (Solution SD-3) has the highest accuracy among these four solutions. However, if Solution SD-2 is selected rather than Solution SD-3, a comparable accuracy with a lower Flash footprint and a lower $\mathbb{E}_{edge}$ is achieved. Let us define the detection error as one minus the accuracy of seizure detection. According to Fig. 5(a), the architecture with edge energy consumption of 0.091 mJ (Solution SD-1), illustrated in blue color, has the smallest area occupancy. Hence, for Flash-sensitive and energy-sensitive IoT devices, Solution SD-1 provides a favorable trade-off between the accuracy and resource-usage for seizure detection in the edge energy scenario.

2) Arrhythmia Classification: Next, we investigate the performance of the proposed energy-aware NAS in the edge energy scenario for arrhythmia classification considering wearable systems. Table II shows the prediction performance (accuracy), edge energy, Flash, and RAM for four solutions, each obtained by one energy-aware NAS subjected to its edge energy consumption limit. As the edge energy consumption $\mathbb{E}_{edge}$ increases, the accuracy also increases gradually. The edge energy consumption of 0.270 mJ (Solution AC-3) has the highest accuracy among these four solutions. The accuracy may be improved with the higher edge energy but lower Flash (Solution AC-0 to Solution AC-1) or lower RAM (Solution AC-2 to Solution AC-3). Let us define the classification error as one minus the accuracy of arrhythmia classification. According to Fig. 5(b), the architecture with edge energy consumption of 0.253 mJ (Solution AC-2), illustrated in green color, has an accuracy of 83.8% with lower Flash footprint and energy consumption compared to Solution AC-3. Hence, for Flash-sensitive IoT

TABLE II
Energy-Aware NAS in the Edge Energy Scenario for Seizure Detection (SD) and Arrhythmia Classification (AC)

Application	Solution	$\mathbb{E}_{edge}$ (mJ)	Flash (KB)	RAM (KB)	Accuracy(%)
Seizure Detection	SD-0	0.059	0.3	8.1	59.3
	SD-1	0.091	0.3	8.5	84.1
	SD-2	0.212	1.3	9.5	86.0
	SD-3	0.375	2.5	9.5	87.3
Arrhythmia Classification	AC-0	0.065	0.6	8.1	19.9
	AC-1	0.096	0.1	8.2	39.5
	AC-2	0.253	1.2	8.5	83.8
	AC-3	0.270	4.0	8.1	88.0

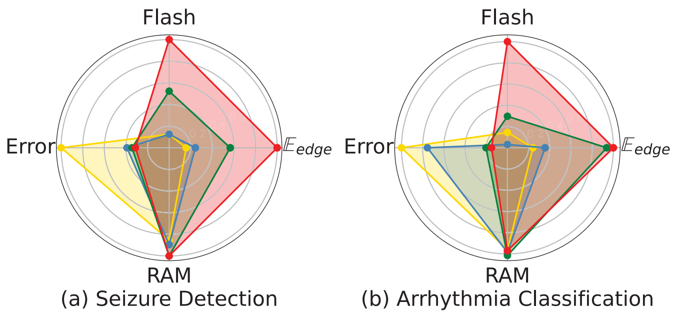


Fig. 5. Normalized comparison of the Flash footprint, RAM footprint, edge energy consumption and error for seizure detection (SD-0: yellow, SD-1: blue, SD-2: green, SD-3: red) and arrhythmia classification (AC-0: yellow, AC-1: blue, AC-2: green, AC-3: red).

TABLE III
ENERGY-AWARE NAS IN THE TOTAL ENERGY SCENARIO FOR SEIZURE DETECTION AND ARRHYTHMIA CLASSIFICATION CONSIDERING DIFFERENT ENERGY WEIGHTS

Application	Factors	Solution	$\mathbb{E}_{total}$(mJ)	$\mathbb{E}_{edge}^{comp}$(mJ)	$\mathbb{E}_{edge}^{comm}$(mJ)	$\mathbb{E}_{fog}$(mJ)	Accuracy(%)
Seizure Detection	10	SD-1	0.103	0.071	0.020	0.012	84.1
		SD-2	0.281	0.205	0.008	0.068	86.0
		SD-3	0.382	0.367	0.008	0.007	87.3
	2	SD-1	0.151	0.071	0.020	0.060	84.1
		SD-2	0.552	0.205	0.008	0.339	86.0
		SD-3	0.410	0.367	0.008	0.035	87.3
	1	SD-1	0.192	0.191	0.001	0	84.1
		SD-2	0.884	0.883	0.001	0	86.0
		SD-3	0.439	0.438	0.001	0	87.3
Arrhythmia Classification	10	AC-1	0.168	0.055	0.041	0.072	39.5
		AC-2	0.461	0.238	0.015	0.208	83.8
		AC-3	0.730	0.249	0.021	0.460	88.0
	2	AC-1	0.454	0.055	0.041	0.358	39.5
		AC-2	1.294	0.238	0.015	1.041	83.8
		AC-3	2.569	0.249	0.021	2.299	88.0
	1	AC-1	0.775	0.772	0.003	0	39.5
		AC-2	2.323	2.320	0.003	0	83.8
		AC-3	4.851	4.848	0.003	0	88.0

devices, Solution AC-2 is chosen as the optimized architecture for arrhythmia classification in the edge energy scenario. Otherwise, for Flash-insensitive IoT devices, Solution AC-3 is adopted as the optimized architecture with a significantly better prediction performance at the price of only a minor increase in the required energy.

B. Total Energy Scenario

Different fog nodes consume different energy on the same DNN architectures. To simulate the energy consumption on fog nodes, we consider various energy-efficiency factors for different fog nodes ($\mathbb{E}_{fog}^{comp} = \frac{\mathbb{E}_{edge}^{comp}}{factor}$). The energy-efficiency factor captures how efficient the fog nodes are in terms of energy consumption, when compared to the edge nodes. For instance, an energy-efficiency factor of $factor = 10$ indicates that the fog node performs the same processing 10 times more efficiently in terms of energy, compared to the edge nodes. Table III shows the performance of the optimization problem in the total energy scenario considering different energy-efficiency factors. For each energy-efficiency factor, three individual evaluations are conducted. Architectures from Solution SD-1, SD-2 and SD-3 in Table II are borrowed for seizure detection. Architectures from Solution AC-1, AC-2 and AC-3 in Table II are borrowed for arrhythmia classification. For each solution, the same architecture but with a different partition point is optimized considering different energy-efficiency factors.

1) Seizure Detection: As shown in Fig. 6(a), for all three solutions, as the energy-efficiency factor decreases from $factor = 10$ to $factor = 1$, the total energy $\mathbb{E}_{total}$ consumed increases. If the energy-efficiency factor decreases from $factor = 10$ to $factor = 2$, the partition point is fixed and $\mathbb{E}_{edge}$ remains the same. The blue, green, and red areas increase, and the total energy $\mathbb{E}_{total}$ and the energy consumed by the fog $\mathbb{E}_{fog}$ increase. However, for the energy-efficiency factor $factor = 1$, $\mathbb{E}_{fog}$ is always 0 mJ. This is because, for the energy-efficiency factor $factor = 1$, the fog node consumes the same energy as the edge node for the same DNN and the entire DNN runs on the edge node. Offloading the classification output makes total energy the smallest among all the partitioning solutions.

2) Arrhythmia Classification: Similarly, as shown in Fig. 6(b), if the energy-efficiency factor decreases from $factor = 10$ to $factor = 1$, the total energy consumption $\mathbb{E}_{total}$ also increases gradually. If the energy-efficiency factor decreases from $factor = 10$ to $factor = 2$, $\mathbb{E}_{edge}$ remains the same because the partition point does not change. However, the total energy $\mathbb{E}_{total}$ and the energy consumed by the fog $\mathbb{E}_{fog}$ increase. In particular, for the energy-efficiency factor $factor = 1$, the searched whole DNN only runs on the edge node. $\mathbb{E}_{edge}$ becomes higher, and $\mathbb{E}_{fog}$ is always 0 mJ. In this case, we only offload the classification output from edge to fog to make $\mathbb{E}_{total}$ the smallest.

In summary, for fog nodes that have an energy-efficiency factor $factor > 1$, our proposed energy-aware NAS generally partitions the searched DNN and offloads the partial DNN from edge nodes to fog nodes to reduce the total energy consumption. For the fog nodes that have an energy-efficiency factor $factor \leq 1$, our framework deploys all the searched

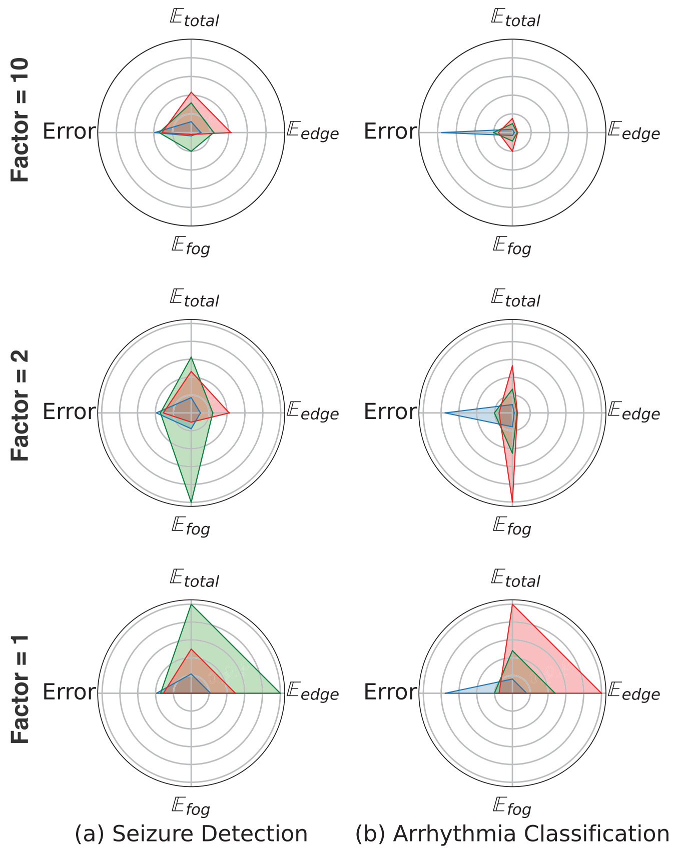


Fig. 6. Normalized comparison of the total energy consumption, the energy consumption on edge nodes and fog nodes, and error for seizure detection (SD-1: blue, SD-2: green, SD-3: red) and arrhythmia classification (AC-1: blue, AC-2: green, AC-3: red).

DNNs on edge nodes to make the total energy consumption the smallest.

C. Ablation Study

To show the importance of core components of our energy-aware NAS framework, the ablation study is conducted. In this subsection, the effectiveness of layer optimization (Section III-B1), the DNN computation partitioning/offloading (Section III-B2), and the comparison between the representative DNNs and the searched DNNs are presented.

1) Layer Optimization: Our energy-aware NAS has the capacity of adjusting the number of layers. For seizure detection, the accuracy is improved from 82.8% to 87.3% if the layer optimization is exploited. For arrhythmia classification, the accuracy is increased from 20.2% to 88.0% if the layer optimization is exploited.

2) DNN Computation Partitioning/Offloading: In this experiment, we partition the searched DNN architecture and offload the partial DNN from edge to fog. As shown in Fig. 7, for a 6-layer DNN (Solution SD-3 in Table II), we calculate the Flash footprint, RAM footprint, edge energy consumption $\mathbb{E}_{edge}$ ($\mathbb{E}_{edge} = \mathbb{E}_{edge}^{comp} + \mathbb{E}_{edge}^{comm}$) for each partition point. Specifically, at the first partition point, the input data is directly transmitted from edge to fog. Therefore, the Flash occupies 0 Bytes (B) and the $\mathbb{E}_{edge}^{comp}$ is 0 mJ. The edge energy consumption $\mathbb{E}_{edge}$ is equal to $\mathbb{E}_{edge}^{comm}$, which is 1.371 mJ. The consumed resources vary at different partition points. If we increase the partition point, the Flash footprint and $\mathbb{E}_{edge}^{comp}$ increase gradually because more parts of the searched DNN run on the edge node. However, the RAM footprint tends to be stable because it is only decided by the maximum memory footprint of neurons between every two layers. The RAM footprint reaches its maximum value at partition point 2. As for the $\mathbb{E}_{edge}^{comm}$, it depends on the length and the channel of the partitioned layer. As illustrated in Fig. 7, at partition point 4, the edge energy consumption $\mathbb{E}_{edge}$ reaches its minimum value.

As shown in Table IV, four DNN architectures are selected from Table II and the comparison of saved edge energy consumption is illustrated. More specifically, for seizure detection, the edge energy consumption of Solution SD-1 "without partition" is 2.1 times that of "with partition", and the edge energy consumption of Solution SD-2 "without partition" is 4.2 times that of "with partition". For arrhythmia classification, the edge energy consumption of Solution AC-1 "without partition" is 8.1 times the edge energy consumption of "with partition". Similarly, the edge energy consumption of Solution AC-2 "without partition" is 9.2 times the edge energy consumption of "with partition".

3) Representative DNNs: Although the DNN computation partitioning has been used in previous works [28], [29], [30], [31], [37], [38], [39], they only partition representative DNNs such as VGG [45]. Our proposed framework considers heterogeneous DNNs for specified hardware devices and distributed IoT systems, which jointly optimizes DNN architectures and DNN computation partitioning/offloading for the first time. The standard VGG11 [45] has 133 million parameters. To show the comparison, for EEG data (seizure detection) and ECG data (arrhythmia classification), we adopt a variant of one-dimensional VGG11. This variant has 34 million parameters, which is around a quarter of the standard VGG11. We consider this variant of one-dimensional VGG11 in our ablation study.

We conduct three experiments including VGG11 without partition, VGG11 with partition, and our proposed framework. The comparison of edge energy consumption $\mathbb{E}_{edge}$, total energy consumption $\mathbb{E}_{total}$, Flash footprint, RAM footprint, and accuracy are shown in Table V. For the VGG11 without partition, the edge energy consumption is 2220.821 mJ, and the total energy consumption is also 2220.821 mJ (the energy-efficiency factor $factor = 1$ is exploited herein for simplification). When VGG11 is exploited, the accuracy of seizure detection and arrhythmia classification is 78.7% and 92.1%, respectively. However, the Flash footprint is 130570.0 KB and the RAM footprint is 256.0 KB, which exceeds the capacity of typical IoT devices with STM32L476RG (ARM Cotex-M4) and similar microcontrollers based on ARM Cortex-M family. Besides, for the VGG11 with partition, the total energy consumption is still 2220.821 mJ, not satisfying the allocated energy budget of IoT devices.

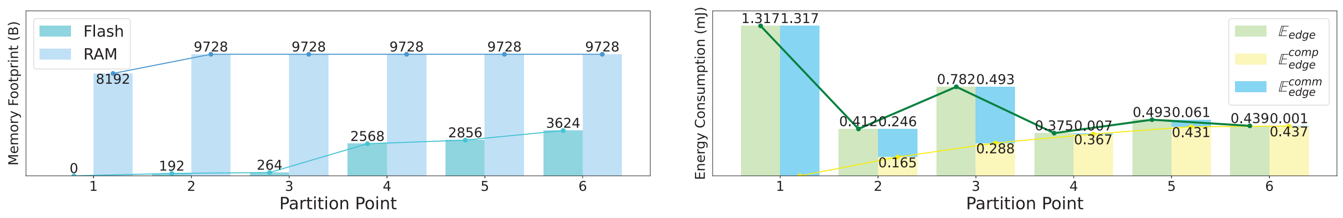


Fig. 7. Different partition points lead to different resource overheads. The sub-optimization problem minimizes the energy consumption of the same architecture.

TABLE IV
SEARCHED DNN ARCHITECTURES AND SAVED ENERGY CONSUMPTION

Solution	Length (Partition Point)	Channel	W/o Partition (mJ)	With Partition (mJ)	Application
SD-1	[2048,128,8,64,2] (3)	[1,1,4,2,1]	0.192	0.091	Seizure Detection
SD-2	[2048,128,4,128,2] (3)	[1,3,3,2,1]	0.884	0.212	Seizure Detection
AC-1	[2048,64,256,512,4,16,16,256,5] (2)	[1,1,1,3,2,4,4,1,1]	0.775	0.096	Arrhythmia Classification
AC-2	[2048,32,8,128,256,1024,5] (3)	[1,4,3,4,4,2,1]	2.323	0.253	Arrhythmia Classification

TABLE V
COMPARISON BETWEEN REPRESENTATIVE DNNS AND SEARCHED DNNS BY ENERGY-AWARE NAS

Setup	VGG11 Without Partition		VGG11 With Partition		Our Framework	
	Seizure Detection	Arrhythmia Classification	Seizure Detection	Arrhythmia Classification	Seizure Detection	Arrhythmia Classification
Solution	A variant of one-dimensional VGG11		A variant of one-dimensional VGG11		SD-3	AC-3
$\mathbb{E}_{edge}$(mJ)	2220.821		1.317		0.375	0.270
$\mathbb{E}_{total}$(mJ)	2220.821		2220.821		0.439	4.851
Flash(KB)	130570.0		0.0		2.5	4.0
RAM(KB)	256.0		8.0		9.5	8.1
Accuracy(%)	78.7	92.1	78.7	92.1	87.3	88.0

In our proposed framework, the joint optimization aims at optimizing the DNN architectures and DNN computation partitioning/offloading. When compared to the VGG11 with partition, the edge energy consumption of our proposed framework is decreased by a factor of 3.5 times (from 1.317 mJ to 0.375 mJ). In addition, the total energy consumption of our framework is decreased by a factor of 5058.8 times (from 2220.821 mJ to 0.439 mJ) for seizure detection. The accuracy of seizure detection in our framework is 87.3%, which is higher than 78.7% of VGG11. Besides, when compared to the VGG11 without partition, for arrhythmia classification, the edge energy consumption of our framework is decreased by a factor of 8225.3 times (from 2220.821 mJ to 0.270 mJ). Moreover, the total energy consumption of our framework is decreased by a factor of 457.8 times (from 2220.821 mJ to 4.851 mJ). The price to pay for making this trade-off, however, is a marginal loss in terms of the accuracy of arrhythmia classification. These results show that the joint optimization of DNN architectures and DNN computation partitioning/offloading enables avoiding the suboptimal DNNs for distributed IoT systems.

D. NAS Benchmarks

We have also evaluated our framework against the state-of-the-art NAS benchmarks on the two real-world medical IoT applications, namely, seizure detection and arrhythmia classification. In this experiment, several NAS schemes, such as NASNet [49] (Reinforcement Learning), DARTS [50] (Gradient-Based), SPOS [17] (One-Shot), CNN-GA [53] (Evolutionary Algorithms), and training-free NAS (Zero-Shot NAS [54]) were considered. All these NAS schemes solely focus on optimizing DNN architectures, without considering DNN computation partitioning/offloading jointly in the optimization. Table VI presents the comparison between our NAS framework and NAS benchmarks on seizure detection and arrhythmia classification.[1]

a) NASNet [49] (Reinforcement Learning): Our NAS framework demonstrates a marginally higher accuracy in seizure detection, and a comparable accuracy in arrhythmia classification, when compared against NASNet [49]. Regarding the searched architectures' complexity, our NAS framework exhibits a substantial reduction, boasting 47 times fewer parameters, 273 times fewer FLOPs in seizure detection, and 51 times fewer parameters, 18 times fewer FLOPs in arrhythmia classification, compared to NASNet [49].

[1]NASNet [49]: https://github.com/MarSaKi/nasnet
DARTS [50]: https://github.com/quark0/darts
SPOS [17]: https://github.com/ShunLu91/Single-Path-One-Shot-NAS
CNN-GA [53]: https://github.com/Marius-Juston/AutoCNN

TABLE VI
COMPARISON BETWEEN OUR NAS FRAMEWORK AND NAS BENCHMARKS

NAS Benchmarks	Types	(Partitioning & Offloading)	Seizure Detection			Arrhythmia Classification		
			Accuracy	Parameters	FLOPs	Accuracy	Parameters	FLOPs
NASNet [49]	Reinforcement Learning	✘	84.3	15.9K	4.09M	89.8	14.9K	0.55M
DARTS [50]	Gradient-Based	✘	77.7	382K	121M	94.8	346K	15.7M
SPOS [17]	One-Shot	✘	81.5	6923K	352M	92.9	6923K	46.9M
CNN-GA [53]	Evolutionary Algorithms	✘	79.6	914K	1612M	91.5	3240K	1061M
Zero-Shot (#Params) [54]	Evolutionary Algorithms	✘	86.6	2.48K	0.104M	88.9	3.79K	0.54M
Zero-Shot (#FLOPs) [54]	Evolutionary Algorithms	✘	85.3	1.47K	0.124M	82.4	1.32K	0.62M
Ours without Partitioning	Evolutionary Algorithms	✘	87.3	0.54K	0.017M	89.7	1.67K	0.11M
Ours (with Partitioning)	Evolutionary Algorithms	✔	87.3	0.34K	0.015M	89.7	0.29K	0.03M

b) DARTS [50] (Gradient-Based): Our NAS framework attains a significantly higher accuracy in seizure detection, albeit with a slightly lower accuracy in arrhythmia classification, compared to DARTS [50]. In terms of complexity, we observe a significant reduction in the parameters of our NAS framework, decreasing by a factor of 1124 times in seizure detection, and by a factor of 1193 times in arrhythmia classification, in comparison to DARTS [50]. In addition, the FLOPs of our NAS framework remarkably decrease by a factor of 8067 times in seizure detection, and by a factor of 523 times in arrhythmia classification, compared to DARTS [50].

c) SPOS [17] (One-Shot): Our NAS framework achieves a higher accuracy in seizure detection, and a lower accuracy in arrhythmia classification, in comparison to SPOS [17]. Notably, concerning parameters, our NAS framework attains a 20362-fold decrease in seizure detection, and a 23872-fold decrease in arrhythmia classification, compared to SPOS [17]. Regarding FLOPs, our NAS framework demonstrates a noteworthy 23467-fold decrease in seizure detection, and a 1563-fold decrease in arrhythmia classification, compared to SPOS [17].

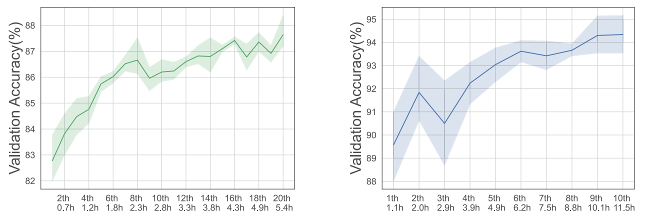
d) CNN-GA [53] (Evolutionary Algorithms): Our NAS framework exhibits a significantly higher accuracy in seizure detection, and a slightly lower accuracy in arrhythmia classification, compared to CNN-GA [53]. Our NAS framework attains remarkable reductions in terms of architecture complexity, featuring 2688 times fewer parameters and 107467 times fewer FLOPs in seizure detection and 11172 times fewer parameters and 35367 times fewer FLOPs on arrhythmia classification, compared to CNN-GA [53].

e) Training-free NAS [83] (Zero-Shot NAS [54]): Zero-shot NAS exploits proxies that directly predict the classification performance of the DNN architectures without training the search architectures. The quantitative comparison among various proxies in Zero-Shot NAS [54] indicates that the proxies quantifying the number of parameters (#Params) and the floating point operations per second (#FLOPs) are not necessarily worse than other well-designed zero-shot proxies [84], [85], [86] in terms of the correlation between these proxies and the real test accuracy; these proxies #Params and #FLOPs generally achieve comparable test accuracy to other proxies, and even better test accuracy in certain datasets. The potential explanation for why #Params and #FLOPs work is that more parameters capture a higher expressive capacity [87] and higher generalization capacity [88] of the architectures. Therefore, we choose #Params and #FLOPs as the proxies in Zero-Shot NAS for evaluating our NAS framework.

Our NAS framework achieves higher accuracy and fewer parameters and FLOPs in seizure detection and arrhythmia classification, compared to Zero-Shot NAS [54]. In terms of parameters, our NAS framework reduces the number of parameters by a factor of 7.3 times in seizure detection, and by a factor of 13.1 times in arrhythmia classification, in comparison to Zero-Shot (#Params) [54]. In terms of FLOPs, we observe a significant reduction in the FLOPs of our NAS framework, decreasing by a factor of 8.3 times in seizure detection, and by a factor of 20.7 times in arrhythmia classification, in comparison to Zero-Shot (#FLOPs) [54]. However, Zero-Shot NAS consumes less time and energy because they require no training. For Zero-Shot NAS with the proxy #Params, the average consumed time for each generation is 20 seconds and 4 seconds for seizure detection and arrhythmia classification, respectively. For Zero-Shot NAS with the proxy #FLOPs, the average consumed time for each generation is 112 seconds and 8 seconds for seizure detection and arrhythmia classification, respectively. Instead, our NAS framework consumes 972 seconds and 4140 seconds for seizure detection and arrhythmia classification, respectively.

E. Evolutionary Process

To demonstrate the effectiveness of our proposed NAS framework, we investigated the evolutionary process during the training phase for seizure detection and arrhythmia classification. As illustrated in Fig. 8, 5.4 GPU hours are needed for seizure detection and 11.5 GPU hours are needed for arrhythmia classification. Despite the cost of the training process, the relationship between evolutionary validation accuracy and evolutionary generation shows that our carefully designed joint optimization tends to reach high accuracy in the early generation stages. Furthermore, our proposed joint optimization is by no means restricted to the evolutionary algorithms and can be seamlessly adopted by other NAS schemes, such as Reinforcement Learning (RL)-based NAS [49], Gradient-based NAS [50], One-Shot NAS [17], Meta-learning NAS [51], Bayesian Optimized NAS [52], and Zero-Shot NAS [54], [83].



(a) Seizure Detection (b) Arrhythmia Classification

Fig. 8. Evolutionary validation accuracy vs. evolutionary generation in the training phase.

VI. Conclusion

In this paper, we propose an energy-aware NAS for distributed IoT, which searches for DNNs with the optimized prediction performance subjected to constrained resources. Our framework jointly optimizes DNN architectures and DNN computation partitioning/offloading for the first time. We have evaluated our energy-aware NAS framework based on two real-world medical IoT applications with wearable technologies, namely, seizure detection [48] and arrhythmia classification [55]. The evaluation results show that with our proposed energy-aware NAS, the accuracy of seizure detection reaches 87.3% with the edge energy consumption of 0.375 mJ. The accuracy of arrhythmia classification reaches 88.0% with the edge energy consumption of 0.270 mJ. In addition, our framework decreases the edge energy consumption by 8225.3 times and the total energy consumption by 5058.8 times compared to the representative VGG11. Furthermore, compared to NAS benchmarks, our proposed NAS framework achieves a more lightweight DNN architecture while maintaining accuracy levels comparable. Finally, our proposed NAS framework can be seamlessly incorporated into other NAS methods [17], [49], [50], [51], [52], [53], [54], enhancing its adaptability and versatility. Our framework is orthogonal to NAS acceleration [89], on-device learning [90], compact search space design [47], [91], [92], DNNs compression [93], [94], [95], and can be combined and powered by these techniques to make the design space exploration faster, DNNs more compact, and energy more efficient.

References

[1] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, “Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare,” *Future Gener. Comput. Syst.*, vol. 78, pp. 659–676, 2018.

[2] D. Sopic, A. Aminifar, A. Aminifar, and D. Atienza, “Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 5, pp. 982–992, Oct. 2018.

[3] J. Wannenburg and R. Malekian, “Body sensor network for mobile health monitoring, a diagnosis and anticipating system,” *IEEE Sensors J.*, vol. 15, no. 12, pp. 6839–6852, Dec. 2015.

[4] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fog computing in healthcare Internet of Things: A case study on ECG feature extraction,” in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, Piscataway, NJ, USA: IEEE, 2015, pp. 356–363.

[5] T. N. Gia et al., “Low-cost fog-assisted health-care IoT system with energy-efficient sensor nodes,” in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Piscataway, NJ, USA: IEEE, 2017, pp. 1765–1770.

[6] A. M. Rahmani et al., “Exploiting smart e-health gateways at the edge of healthcare Internet-of-Things: A fog computing approach,” *Future Gener. Comput. Syst.*, vol. 78, pp. 641–658, 2018.

[7] P. Perego, *Device for mHealth*. Cham: Springer International Publishing, 2019, pp. 87–99. [Online]. Available: https://doi.org/10.1007/978-3-030-02182-5_6

[8] F. Forooghifar, A. Aminifar, and D. Atienza, “Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1338–1350, Dec. 2019.

[9] E. De Giovanni, A. A. ValdÉs, M. PeÓn-QuirÓs, A. Aminifar, and D. Atienza, “Real-time personalized atrial fibrillation prediction on multi-core wearable sensors,” *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1654–1666, Oct.–Dec. 2021.

[10] R. Zanetti, A. Arza, A. Aminifar, and D. Atienza, “Real-time EEG-based cognitive workload monitoring on wearable devices,” *IEEE Trans. Biomed. Eng.*, vol. 69, no. 1, pp. 265–277, Jan. 2022.

[11] S. Baghersalimi, T. Teijeiro, D. Atienza, and A. Aminifar, “Personalized real-time federated learning for epileptic seizure detection,” *IEEE J. Biomed. Health Inform.*, vol. 26, no. 2, pp. 898–909, Feb. 2022.

[12] S. Baghersalimi, T. Teijeiro, A. Aminifar, and D. Atienza, “Decentralized federated learning for epileptic seizures detection in low-power wearable systems,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 6392–6407, May 2024.

[13] A. Aminifar, M. Shokri, and A. Aminifar, “Privacy-preserving edge federated learning for intelligent mobile-health systems,” *Future Gener. Comput. Syst.*, vol. 161, pp. 625–637, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24003972

[14] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, “A survey on evolutionary neural architecture search,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 237–285, May 1996.

[16] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in *Proc. Int. Conf. Learn. Represent.*, 2018.

[17] Z. Guo et al., “Single path one-shot neural architecture search with uniform sampling,” in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Part XVI, 16. Heidelberg, Germany: Springer, Aug. 23–28, 2020, pp. 544–560.

[18] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, “A comprehensive survey on hardware-aware neural architecture search,” Ph.D. dissertation, LAMIH, Valenciennes, France: Université Polytechnique des Hauts-de-France, 2021.

[19] K. T. Chitty-Venkata and A. K. Somani, “Neural architecture search survey: A hardware perspective,” *ACM Comp. Surv.*, vol. 55, no. 4, pp. 1–36, Nov. 2022. [Online]. Available: https://doi.org/10.1145/3524500

[20] Y. Jiang, X. Wang, and W. Zhu, “Hardware-aware transformable architecture search with efficient search space,” in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Piscataway, NJ, USA: IEEE, 2020, pp. 1–6.

[21] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, “MCUNet: Tiny deep learning on IoT devices,” in *Proc. Adv. Neural Inf. Process. Syst.*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Red Hook, NY, USA: Curran Associates, Inc., 2020, pp. 11711–11722. Accessed: Oct. 10, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/86c51678350f656dcc7f490a43946ee5-Paper.pdf

[22] X. Wang, M. Magno, L. Cavigelli, and L. Benini, “FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things,” *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4403–4417, May 2020.

[23] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, “Memory-efficient patch-based inference for tiny deep learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Red Hook, NY, USA: Curran Associates, Inc., 2021, pp. 2346–2358. Accessed: Oct. 10, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/1371bccec2447b5aa6d96d2a540fb401-Paper.pdf

[24] C. Banbury et al., "MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers," *Proc. Mach. Learn. Syst.*, vol. 3, pp. 517–532, 2021.

[25] I. Fedorov, R. P. Adams, M. Mattina, and P. Whatmough, "Sparse: Sparse architecture search for CNNs on resource-constrained microcontrollers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 4977–4989.

[26] E. Liberis, L. Dudziak, and N. D. Lane, "μNAS: Constrained neural architecture search for microcontrollers," in *Proc. 1st Workshop Mach. Learn. Syst.*, 2021, pp. 70–79.

[27] B. Wu et al., "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 10726–10734.

[28] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

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

[30] A. E. Eshratifar and M. Pedram, "Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment," in *Proc. Great Lakes Symp. VLSI*, 2018, pp. 111–116.

[31] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[32] Y. Liu, Q. Chen, G. Liu, H. Liu, and Q. Yang, "EcoSense: A hardware approach to on-demand sensing in the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 37–43, Dec. 2016.

[33] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Inf. Syst.*, vol. 107, 2022, Art. no. 101840.

[34] F. Firouzi, B. Farahani, E. Panahi, and M. Barzegari, "Task offloading for edge-fog-cloud interplay in the healthcare Internet of Things (IoT)," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Piscataway, NJ, USA: IEEE, 2021, pp. 1–8.

[35] J. Zhang, J. Wang, Z. Yuan, W. Zhang, and L. Liu, "Offloading demand prediction-driven latency-aware resource reservation in edge networks," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13826–13836, Aug. 2023.

[36] H. Ma, R. Li, X. Zhang, Z. Zhou, and X. Chen, "Reliability-aware online scheduling for DNN inference tasks in mobile edge computing," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11453–11464, Jul. 2023.

[37] F. Jia et al., "CoDL: Efficient CPU-GPU co-execution for deep learning inference on mobile devices," in *Proc. 20th Annu. Int. Conf. Mobile Syst., Appl. Services*, New York, NY, USA: ACM, 2022, pp. 209–221.

[38] G. Liu et al., "An adaptive DNN inference acceleration framework with end–edge–cloud collaborative computing," *Future Gener. Comput. Syst.*, vol. 140, pp. 422–435, 2023.

[39] X. Liu and A. G. Richardson, "Edge deep learning for neural implants: A case study of seizure detection and prediction," *J. Neural Eng.*, vol. 18, no. 4, 2021, Art. no. 046034.

[40] X. Dai, Z. Xiao, H. Jiang, and J. C. S. Lui, "UAV-assisted task offloading in vehicular edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2520–2534, Apr. 2024.

[41] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000–4015, Jul. 2023.

[42] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[43] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[44] W. Fan et al., "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 238–256, Jan. 2024.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*. San Diego, CA, USA: Computational and Biological Learning Society, 2015.

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

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

[48] A. H. Shoeb, "Application of machine learning to epileptic seizure onset detection and treatment," Ph.D. dissertation, MIT Press, Cambridge, MA, USA, 2009.

[49] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[50] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[51] J. Wang, J. Wu, H. Bai, and J. Cheng, "M-NAS: Meta neural architecture search," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 04, pp. 6186–6193, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6084

[52] C. White, W. Neiswanger, and Y. Savani, "BANANAS: Bayesian optimization with neural architectures for neural architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 10293–10301.

[53] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[54] G. Li, D. Hoang, K. Bhardwaj, M. Lin, Z. Wang, and R. Marculescu, "Zero-shot neural architecture search: Challenges, solutions, and opportunities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 7618–7635, Dec. 2024.

[55] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, May/Jun. 2001.

[56] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 2902–2911.

[57] A. Burrello, M. Risso, B. A. Motetti, E. Macii, L. Benini, and D. J. Pagliari, "Enhancing neural architecture search with multiple hardware constraints for deep learning model deployment on Tiny IoT devices," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 3, pp. 780–794, Jul.–Sep. 2024.

[58] Z. Lu et al., "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, 2019, pp. 419–427.

[59] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Part I, 16. Heidelberg, Germany: Springer, Aug. 23–28, 2020, pp. 35–51.

[60] Y. Xue, C. Chen, and A. Słowik, "Neural architecture search based on a multi-objective evolutionary algorithm with probability stack," *IEEE Trans. Evol. Comput.*, vol. 27, no. 4, pp. 778–786, Aug. 2023.

[61] S. Li, Y. Sun, G. G. Yen, and M. Zhang, "Automatic design of convolutional neural network architectures under resource constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3832–3846, Aug. 2023.

[62] B. Lyu, H. Yuan, L. Lu, and Y. Zhang, "Resource-constrained neural architecture search on edge devices," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 134–142, Jan./Feb. 2022.

[63] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu, and D. Z. Pan, "Mixed precision neural architecture search for energy efficient deep learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, Piscataway, NJ, USA: IEEE, 2019, pp. 1–7.

[64] X. Dai et al., "ChamNet: Towards efficient network design through platform-aware model adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 11390–11399.

[65] C.-H. Hsu et al., "MONAS: Multi-objective neural architecture search using reinforcement learning," 2018, *arXiv:1806.10332*.

[66] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, p. 386, 1958.

[67] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. Int. Conf. Global Trends Signal Process., Inf. Comput. Commun. (ICGTSPICC)*, Piscataway, NJ, USA: IEEE, 2016, pp. 261–265.

[68] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[69] D. Velasco-Montero, J. Fernández-Berni, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "PreVIous: A methodology for prediction of visual inference performance on IoT devices," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9227–9240, Oct. 2020.

[70] H. Cai et al., "Enable deep learning on mobile devices: Methods, systems, and applications," *ACM Trans. Des. Automat. Electron. Syst. (TODAES)*, vol. 27, no. 3, pp. 1–50, 2022.
[71] D. Sopic, A. Aminifar, and D. Atienza, "e-Glass: A wearable system for real-time detection of epileptic seizures," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Piscataway, NJ, USA: IEEE, 2018, pp. 1–5.
[72] B. Huang, R. Zanetti, A. Abtahi, D. Atienza, and A. Aminifar, "EpilepsyNet: Interpretable self-supervised seizure detection for low-power wearable systems," in *Proc. IEEE 5th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Piscataway, NJ, USA: IEEE, 2023, pp. 1–5.
[73] B. Huang, A. Abtahi, and A. Aminifar, "Lightweight machine learning for seizure detection on wearable devices," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–2.
[74] A. Aminifar, B. Huang, A. A. Fahliani, and A. Aminifar, "LightFF: Lightweight inference for forward-forward algorithm," in *Proc. 27th Eur. Conf. Artif. Intell. (ECAI)*, Amsterdam, The Netherlands: IOS Press, 2024, pp. 1728–1735.
[75] D. G. Zill, *Advanced Engineering Mathematics*. Boston, MA, USA: Jones & Bartlett, 2020.
[76] V. Arlington, "Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," Rep. ANSI-AAMI EC57, 1998.
[77] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "ECG heartbeat classification: A deep transferable representation," in *Proc. IEEE Int. Conf. Healthcare Inform. (ICHI)*, Piscataway, NJ, USA: IEEE, 2018, pp. 443–444.
[78] G. Surrel, A. Aminifar, F. Rincón, S. Murali, and D. Atienza, "Online obstructive sleep apnea detection on medical wearable sensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 4, pp. 762–773, Aug. 2018.
[79] N. Semiconductor, "Online power profiler," 2019. Accessed: Oct. 10, 2024. [Online]. Available: https://devzone.nordicsemi.com/power/
[80] A. F. Gad, "PyGAD: An intuitive genetic algorithm Python library," *Multimed Tools Appl.*, vol. 83, pp. 58029–58042, 2024, doi: 10.1007/s11042-023-17167-y.
[81] P. Moritz et al., "Ray: A distributed framework for emerging AI applications," in *Proc. 13th USENIX Symp. Operat. Syst. Des. Implementation (OSDI)*, 2018, pp. 561–577.
[82] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operat. Syst. Des. Implementation (OSDI)*, vol. 16, Savannah, GA, USA, 2016, pp. 265–283.
[83] M.-T. Wu and C.-W. Tsai, "Training-free neural architecture search: A review," *ICT Exp.*, vol. 10, no. 1, pp. 213–231, 2024.
[84] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in *Proc. Int. Conf. Learn. Represent.*, 2021.
[85] N. Lee, T. Ajanthan, and P. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[86] M. Lin et al., "Zen-NAS: A zero-shot NAS for high-performance image recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 337–346.
[87] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning: A survey," *Knowl. Inf. Syst.*, vol. 63, pp. 2585–2619, 2021.
[88] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 6158–6169.
[89] Q. Ye, Y. Sun, J. Zhang, and J. Lv, "A distributed framework for EA-based NAS," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1753–1764, Jul. 2021.
[90] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates Inc., 2020.
[91] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size," 2016, *arXiv:1602.07360*.
[92] 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.*, 2018, pp. 6848–6856.
[93] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, Conf. Track Proc., San Juan, Puerto Rico, May 2–4, 2016.
[94] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.
[95] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 kB RAM for the Internet of Things," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 1935–1944.

Baichuan Huang (Student Member, IEEE) received the bachelor's degree in electronic information engineering and the master's degree from the State Key Laboratory of LIESMARS, Wuhan University, China, in 2021. He is currently working toward the Ph.D. degree with the Wallenberg AI, Autonomous Systems and Software Program, Department of Electrical and Information Technology, LTH, Lund University, Sweden. In 2020, he worked as a Research Intern with Tencent Holdings Ltd., Shanghai, China. In 2019, he worked as a Research Intern with the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. His research interests include tinyML, Internet of Things (IoT), mobile health, and edge artificial intelligence.

Azra Abtahi (Member, IEEE) received the Ph.D. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2018, supported by a scholarship from Iran's National Elites Foundation. In 2016, she participated in a year-long sabbatical program with Queens University, ON, Canada. Following her doctoral studies, she served as a Postdoctoral Fellow with Sharif University of Technology, until 2019. She is currently working as a WASP Postdoctoral Fellow with the Department of Electrical and Information Technology, Lund University. She has worked on various areas of machine learning, Internet of Things, and signal processing. Her research interests include these subjects.

Amir Aminifar (Senior Member, IEEE) received the Ph.D. degree from Swedish National Computer Science Graduate School (CUGS), Linköping University, Sweden, in 2016. He is an Assistant Professor with the Department of Electrical and Information Technology, Lund University, Sweden. From 2014 to 2015, he visited the University of California, Los Angeles (UCLA), USA, and Sant'Anna School of Advanced Studies, Italy. From 2016 to 2020, he held a Scientist position with the Institute of Electrical Engineering, Swiss Federal Institute of Technology (EPFL), Switzerland. His research interests include edge machine learning for Internet of Things (IoT) systems, intelligent mobile-health and wearable systems, and health informatics.