Early-Exit Neural Architecture Search for Energy-Harvesting Edge Computing

Pierre-Louis Sixdenier, Mark Deutel, and Jürgen Teich
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
{pierre-louis.e.sixdenier, mark.deutel, juergen.teich}@fau.de

***Abstract*—Energy harvesting (EH) is increasingly used in wireless sensor nodes to extend the lifetime of battery-powered devices in remote locations. On such devices, the use of deep neural networks (DNNs) is limited not only by the memory and computational constraints of the system, but also by the often large amount of energy required to perform DNN inference. As a result, continuous DNN inference cannot be guaranteed on such systems, especially under severe power constraints resulting from small battery capacities and solar panel sizes. A promising approach to dynamically adjust the power consumption of DNN inference are early-exit neural networks (EE-NNs), that allow a DNN to exit early at different points during its inference. In this paper, we propose a simulation-aided neural architecture search framework to explore EE-DNN architectures (EENAS) and an online energy manager (EM) that dynamically decides which early exit to take based on the available energy and the EE-NN's confidence in its predictions while ensuring energy-neutral operation (ENO). We evaluate our approach on four vision datasets and show not only that our EENAS approach can find EE-NNs with low energy consumption that maintain a high accuracy, but also that our proposed EM can execute the explored EE-NNs with an up to 36% improved accuracy compared to the state-of-the-art while satisfying the ENO constraints.**

***Index Terms*—IoT, Energy Harvesting, Neural Networks**

I. Introduction

Energy harvesting (EH) can be used to extend the lifetime of battery-powered Wireless Sensor Networks (WSNs) running in remote locations as in, e.g., agricultural [1] and environmental monitoring [2]. However, the intermittence of the energy sources makes it difficult to ensure that the nodes in the network operate without interruption. To address this issue, energy management strategies have been proposed [3] to maximize a given quality of service (QoS), e.g., the nodes' duty cycles, or a user-defined accuracy, while satisfying constraints on the nodes' state of charge (SoC) to enable so-called energy-neutral operation (ENO).

Deep Neural Network (DNN) inference is often performed on sensor nodes in WSNs locally to reduce the amount of data that must be transmitted over the network. This local processing and filtering of data is considered essential because wireless communication of all perceived data would quickly drain a node's battery. However, the limited resources of the nodes, such as RAM and ROM, make it difficult to deploy even DNNs of moderate size, making the use of compression techniques such as pruning and quantization essential [4]. If only a certain amount of weights can be stored, one can use weight clustering [5]. Alternatively, the precision of the weights can be reduced to as low as one bit only [6]. Recently, optimization techniques that are able to explore as well the space of network structures as the degree of pruning and quantization so to tradeoff complexity and accuracy have been proposed under the term neural architecture search (NAS), see, e.g., [7], [8].

Early-exit DNNs (EE-DNNs) [9] denote a type of neural network that allow for multiple exits to be taken at different network depths instead of only one exit at the end. This means that an EE-DNN can stop processing at an earlier layer, thereby reducing computation and energy consumption at the cost of potentially less accurate predictions.

In this paper, we exploit the opportunities of EE-DNNs for energy saving in the context of EH-WSNs. Here, a power manager could decide at run time in EH-WSNs, where it allows a power manager to decide which early exit to take based not only on prediction confidence, but also on available energy. In this context, we present a novel simulation-assisted multi-objective neural architecture search method (EENAS) for EE-DNNs that maximizes accuracy, and the computational effort in terms of number of Multiply-Accumulates (MACs), while considering memory constraints on the target platform.

For evaluating the accuracy of an EE-DNN candidate under the tight energy constraints in the context of an EH-WSN, a typical daily harvesting energy profile is applied to a daily schedule of periodic inference activations. This simulative evaluation takes into account also the applied energy management (EM) policy.

Our contributed methodology consists of two major components:

1) The multi-objective static exploration and optimization methodology for EE DNN candidates subject to given energy and device resources constraints, of which we provide an overview in Figure 1 and that we describe in detail in Sec. III-A.
2) The strategy for runtime energy management (EM) by selecting the best early exit of the EE-DNN for each inference while ensuring ENO, which we introduce in Sec. III-B.

We evaluate our approach on four machine learning vision datasets in terms of hypervolume improvement and accuracies obtained for different battery sizes and photovoltaic panel peak powers in Sec. IV.

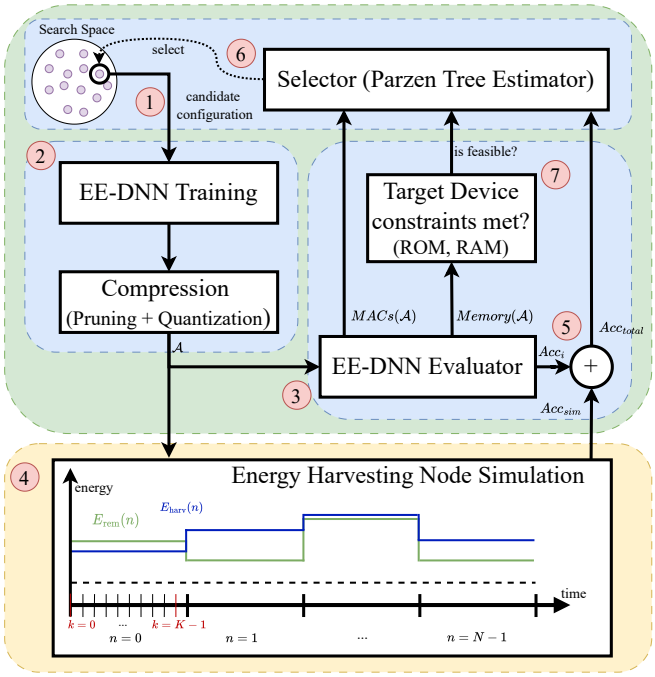


Fig. 1: Proposed multi-objective EENAS methodology (top) and simulative evaluation of accuracy of network candidates (bottom). A candidate EE-DNN configuration is picked from the search space by the selector (1). After applying training and compresssion (2), the resulting candidate $\mathcal{A}$ is then evaluated in terms of its number of Multiply-Accumulates (MACs), memory, and average accuracy Acc_{a} for each of its early exits (3). Additionally, $\mathcal{A}$ is evaluated within an energy harvesting node simulation, where the operation of the EE-DNN on a sensor node when harvesting an amount of energy E_{harv} for one day (24h) is simulated to obtain a total accuracy under simulated conditions Acc_{sim} (4). Acc_{a} and Acc_{sim} are then combined to the objective value Acc_{total} (5). A selector then uses both Acc_{total} and the number of MACs to select the next candidate configuration from the search space (until the search budget is exhausted) (6). The selector thereby also checks whether the memory requirements of $\mathcal{A}$ are within the constraints of the target device (7).

II. Fundamentals

A. Related Work

EE-DNNs [9] were initially proposed to speed up inference by adding additional side classifiers to a DNN. Here, an original backbone DNN is extended by additional sets of layers, the early exit branches, which allow a prediction that is not based on the full feature extractor of the backbone DNN, see Figure 2 for a schematic overview. The points in the backbone DNN where the side classifiers are attached are called anchors.

The benefit of EE-DNNs is, that they allow to exit an inference early for samples where the network is highly confident about the classification, thereby reducing the overall average inference time and computational overhead. Here,

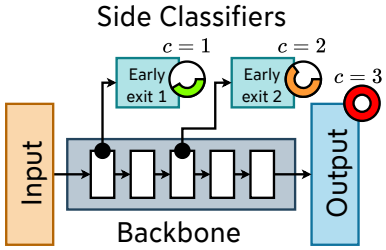


Fig. 2: Example of an EE-DNN architecture candidate $\mathcal{A}$ containing $C = 3$ exits. The backbone network contains the feature extraction layers, of which some are connected to a side classifier at an anchor point (black circles). As the inference proceeds, the input is passed through the backbone network and the side classifiers.

decision of whether an inference should terminate early by taking the intermediate result of a side classifier is based on the classifier's confidence in the current sample. Let $\hat{y}$ be the class probability vector computed by the side classifier over all class labels $l \in L$, its confidence can be computed using the entropy score defined λ as

$$\lambda(\hat{y}) = \sum_{l=0}^{L-1} \hat{y}_l \cdot log(\hat{y}_l). \quad (1)$$

There are different approaches to training EE-DNNs, with the most commonly used being *joint training* [9], [10]. In joint training, fitting the entire DNN, including all of its early exit branches, is treated as a single optimization problem. The loss is computed by either combining the cross-entropy scores of each exit branch [9] or calculating an overall loss from the output of all exit branches combined [10]. Alternatively, *multi-stage training strategies* are used, where first, the backbone network is trained, and then, each early exit branch is trained separately with the backbone frozen [11]. A variant of this approach is to iteratively train every path from the input layer of the DNN to each early exit classifiers as its own DNN with parameters shared with previous iterations [12]. Training the early exit branches separately has also been combined with knowledge distillation, in which the trained backbone network serves as a teacher for the early exit classifiers by calculating the Kullback-Leibler divergence between the probability vectors of them and the backbone network [13].

In this work, we propose a new strategy for training EE-DNNs by combining the training with pruning and quantization within a multi-objective NAS, which allows our approach not only to search for EE-DNNs that maximize accuracy, but also consider resource constraints including memory requirements.

In [14], the authors propose a NAS methodology to optimize the architecture of an EE-DNN for energy-harvesting devices. They also propose an energy management strategy to adaptively select the exit of the EE-DNN to use for each inference, by training a reinforcement learning agent that takes

a decision based on the available energy and the confidence in the output of each early exit. This energy management strategy is combinatorial: it only considers the current state of the sensor node (energy remaining and harvested) but does not plan to be energy-neutral over a longer duration, e.g., an entire day. In contrast, our proposed NAS approach evaluates each candidate for achievable accuracy using a simulative approach shown in Figure 1. Based on an initial energy budget and estimates of harvested energy over a full day, this simulation includes also an energy management component (EM) that decides at each inference which early exit to take based on available energy so to achieve ENO.

B. System Model

We consider a sensor node equipped with a photovoltaic panel and a rechargeable battery. The node harvests energy at a rate assumed constant for each time slot $n \in \{0, \ldots, N-1\}$ of the day (24h), collecting an overall amount of energy $E_{\text{harv}}(n)$ per slot. The harvested energy is used to recharge a battery of capacity E_{max}.

Throughout the day, the node periodically collects, at a period T (with $\frac{24h}{N} = K \cdot T, K \in \mathbb{N}$), data from a set of integrated sensors and performs an inference using a EE-DNN to classify the data. Let the inference of a EE-DNN consisting of C exits, consume the amount of energy $e_{\text{DNN}}(c)$ [J] when taking the exit $c \in \{1, \ldots, C\}$. The case $c = 0$ means not executing the inference at all. This special case must be reserved for preserving energy-neutral operation (ENO) by construction. We assume that the energy consumed for data acquisition, its processing, an energy management policy, data transmission, and possibly staying in sleep mode until the end of the period. Thus, the energy consumed by the DNN during a time slot n can be computed by

$$E_{\text{cons}}(n) = \sum_{c=0}^{C} \sum_{k=0}^{K-1} e_{\text{DNN}}(c) \cdot X_{n,k,c} \tag{2}$$

where $X_{n,k,c} \in \{0, 1\}$ is a decision variable denoting whether the k-th inference of time slot n stops at exit c (i.e., $X_{n,k,c} = 1$) or not (i.e., $X_{n,k,c} = 0$) being the default. Consequently, with E_{harv} denoting the amount of energy harvested during time slot n, the energy remaining in the battery at the end of a time slot n is $E_{\text{rem}}(n+1)$ can be estimated as

$$E_{\text{rem}}(n+1) = \lceil \min\left(E_{\text{rem}}(n) - E_{\text{cons}}(n) + E_{\text{harv}}(n), E_{\text{max}}\right) \rceil^{+} \tag{3}$$

with $\lceil \cdot \rceil^{+} = \max(\cdot, 0)$.

Now, for energy-neutral operation (ENO), the following constraints need to be satisfied: (1) the battery charge never falls below a minimum value E_{min}, i.e.,

$$E_{\text{rem}}(n) \geq E_{\text{min}} \quad \forall n \in \{0, \ldots, N-1\}, \tag{4}$$

and (2) the final battery charge at the end of the day is at least equal to its initial charge, i.e.,

$$E_{\text{rem}}(N) \geq E_{\text{rem}}(0). \tag{5}$$

Finally, for each inference at time slot n and period k, we assume that the EE-DNN produces a class probability vector $\hat{Y}_{n,k,c}$ when using exit c, which aims to predict the ground truth label $Y_{n,k,c}$. The problem of selecting the exit of the EE-DNN to use for each inference in order to minimize the classification errors during the day, while satisfying the ENO constraints, could then be formulated as the following Integer Linear Programming (ILP):

$$\begin{aligned} \max_{X_{n,k,c}} & \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} X_{n,k,c} \cdot \left[\!\left[\hat{Y}_{n,k,c} = Y_{n,k,c} \right]\!\right] \\ \text{s.t.} & \sum_{c=0}^{C} X_{n,k,c} = 1 \\ & \forall\ n \in \{0, \ldots, N-1\}, k \in \{0, \ldots, K-1\} \\ & \text{Eqs. } (2-5) \end{aligned} \tag{6}$$

where $[\![\cdot]\!]$ is the Iverson bracket, i.e., $[\![P]\!] = 1$ if P is true and $[\![P]\!] = 0$ otherwise.

Unfortunately, an optimal solution to the problem shown in Eq. (6) cannot be computed at runtime, as the values of $Y_{n,k,c}$ and $E_{\text{harv}}(n)$ are only available at the end of the time slot n. The time needed to solve the above ILP might also be prohibitive. Therefore, we propose simulation-assisted NAS during the exploration phase, as explained next in Section III-A. Moreover, a simple but effective heuristic for deciding on the exit of each inference is proposed and explained in Section III-B.

III. Methodology

A. Simulation-aided Neural Architecture Search (NAS)

Our NAS methodology consists of three parts marked blue in Figure 1 (top): First, we use Bayesian optimization, i.e., the *Parzen Tree Estimator algorithm* [15], as a selector for EE-DNN candidate selection during exploration. Second, for each selected EE-DNN candidate consiting of a network configuration, we train it on a target dataset and use *iterative L1 norm pruning* [16] and *static quantization* for compression. The use of the compression techniques is controlled by the optimizer via the selected candidate configuration. Third, we evaluate the resulting trained compressed EE-DNN $\mathcal{A}$ by using a runtime environment called *DNNruntime* [17] to (a) map the trained EE-DNNs to C-code, (b) compute their ROM and RAM requirements $Memory(\mathcal{A})$ to validate if the explored EE-DNN is feasible given the constraints of the target MCU, (c) compute the maximum number of MACs $MACs(\mathcal{A})$ required to compute the final exit $c = C$, and (d) the average accuracy Acc_{a} of all exits $c \in \{1, \ldots, C\}$ of the EE-DNN obtained over the evaluation dataset $\hat{a}_c$,

$$Acc_{\text{a}} = \frac{1}{C} \sum_{c=1}^{C} \hat{a}_c.$$

In addition, and to better reflect the performance of the EE-DNNs when deployed on an energy harvesting device, we combine Acc_{a} with the accuracy obtained through an

energy harvesting simulation (see Figure 1, bottom), according to the model defined in Section II-B of the target energy harvesting device, given a harvested energy profile E_{harv} and the characteristics of a sensor node (i.e., battery capacity E_{max}, sampling period T). The accuracy objective Acc_{total} defined in the following thereby encodes not only the accuracy of each exit, but also how well the EE-DNNs can be used in a real-world scenario under given energy constraints:

$$Acc_{\text{total}} = (1-\tau)Acc_{\text{a}} + \tau \cdot Acc_{\text{sim}} \tag{7}$$

where

$$Acc_{\text{sim}} = \sum_{n=0}^{N-1}\sum_{k=0}^{K-1} X_{n,k,c} \cdot \left[\!\left[\hat{Y}_{n,k,c} = Y_{n,k,c}\right]\!\right]$$

is the accuracy obtained by the simulation of the energy harvesting device and τ is a trade-off hyperparameter.

B. Energy Management

The Energy Manager (EM) must perform three types of tasks to be executed at different timescales over a day:

- *Energy-neutral Budget Computation*: At the start of each simulated day, an algorithm computes the maximal energy budget $E_{\text{b}}(n)$ for each time slot n while satisfying ENO is computed. This will be achieved by solving a dynamic programming problem formulated in Eq. (8).
- *Maximal Early-Exit Scheduling*: At the start of each time slot n, a schedule is computed determining for each inference $k, \forall k \in \{0, K-1\}$ within the current time slot n a highest potential exit $c_{\text{max}}(k)$ that might be taken for not violating $E_{\text{b}}(n)$. The corresponding knapsack problem to be solved is introduced in Eq. (9).
- *Entropy-Aware Online Policy*: For each inference k, a runtime policy that, given $c_{\text{max}}(k)$, selects the exit branch of the EE-DNN to use based on the confidence of the DNN in its output. This policy will be introduced in Algorithm 1.

A detailed description of these tasks is given in the following.

1) Energy-neutral Budget Computation: The first task has to be computed once at the beginning of the day, taking as input a trace of predicted energy harvested per time slot $\bar{E}_{\text{harv}}(n)$ and computing an energy-neutral budget $E_{\text{b}}(n)$ for each time slot n. Such an allocation of allowed energy budgets $E_{\text{b}}(n)$ for each time slot n can be computed by solving the following optimization problem using a dynamic programming algorithm:

$$\begin{aligned} \max_{E_{\text{b}}} & \sum_{n=0}^{N-1} E_{\text{b}}(n) \\ \text{s.t. } & E_{\text{rem}}(n+1) = \lceil \min(E_{\text{rem}}(n) - E_{\text{b}}(n) \\ & \quad + \bar{E}_{\text{harv}}(n), E_{\text{max}}) \rceil^{+} \\ & \text{Eqs.}(4-5). \end{aligned} \tag{8}$$

A solution to the problem in Eq. (8) will allow for an ENO over a full day subject that the estimations of harvested energy $\bar{E}_{\text{harv}}(n)$ are exact ($\bar{E}_{\text{harv}}(n) = E_{\text{harv}}(n)\ \forall n \in \{0, \ldots, N-1\}$) as well as the energy estimations in Eq. (2) and Eq. (3).

2) Maximal Early-Exit Scheduling: At the beginning of each time slot n, the EM computes, for each inference $k \in \{0, \ldots, K-1\}$, a latest exit c that can be taken in order to not exceed the overall energy budget $E_{\text{b}}(n)$, given its energy consumption $e_{\text{DNN}}(c)$ and accuracy $\hat{a}_c$. The values of $e_{\text{DNN}}(c)$ for all the exits c of an EE-DNN are modeled for the targeted MCU by the number of MACs required to compute the output of exit c (an example of such a model is given in Section IV-A2). This maximal (allowed) exit schedule can be obtained by solving the following unbounded knapsack problem (UKP)

$$\begin{aligned} \max_{x_{n,k,c}} & \sum_{k=0}^{K-1}\sum_{c=0}^{C} x_{n,k,c} \cdot (1 + \log_{10}(\hat{a}_c)) \\ \text{s.t. } & \sum_{k=0}^{K-1}\sum_{c=0}^{C} e_{\text{DNN}}(c) \cdot x_{n,k,c} \leq E_{\text{b}}(n) \\ & \sum_{c=0}^{C} x_{n,k,c} = 1, \forall k \in \{0, \ldots, K-1\} \end{aligned} \tag{9}$$

with $x_{n,k,c}$ being the decision variable that is 1 ($x_{n,k,c} = 1$) for the highest exit c that can be taken at inference k of time slot n, and zero else. This problem can be solved at the start of each time slot n using also a dynamic programming algorithm [18] with a pseudo-polynomial execution time.

3) Entropy-Aware Online Policy: A simple heuristic for runtime decision-making on if and how to execute the kth inference in time slot n is as follows: If the solution of Eq. (9) delivers $x_{n,k,0} = 1$, then there is not enough energy to execute the DNN at all at this time instant (algorithm 1). Else, we start executing the DNN from $c = 1$ until (whatever comes first) the entropy estimate λ_c is higher than a threshold th_c, or the maximum value c_{max} computed for n and k in algorithm 1 from the solution of the ILP in Eq. (9). If the entropy λ_c of the output layer of exit c is above a pre-computed threshold th_c (algorithm 1), the inference is halted here (algorithm 1). Else, if this is the last exit $c_{\text{max}}(k)$ that can be taken without violating the energy budget, the inference is also stopped algorithm 1. Otherwise, the next exit is computed.

Algorithm 1 Entropy-Aware Online Policy

Input: n, k
Output: $X_{n,k,c}$
if $(x_{n,k,0} = 0)$ **then** ▷ If at least one exit can be scheduled
  $c_{\text{max}} \leftarrow \text{argmax}_c \{x_{n,k,c} \mid x_{n,k,c} = 1\}$
  for $c = 1$ **to** $c_{\text{max}}(k)$ **do**
    Run inference of exit c on the input n, k
    $\lambda_c \leftarrow$ entropy of value at exit c
    if $(\lambda_c > th_c)$ **then**
      $X_{n,k,c} \leftarrow 1$ ▷ The network is confident
      break
    end if
  end for
  $X_{n,k,c_{\text{max}}(k)} \leftarrow 1$ ▷ Otherwise, use the worse-case exit
end if

The threshold values $th_c, \forall c \in \{1, \ldots, C\}$ can be obtained offline using a Bayesian multi-objective optimization algorithm [15] to minimize the amount of energy spent, while maximizing the accuracy obtained when applying the entropy threshold filter.

IV. Experiments

In this section, we first present results on the performance of our EENAS in terms of achieved accuracy, number of MACS and memory requirements of the obtained architectures. Then, we evaluate the simulation accuracy obtained by our energy management in an end-to-end simulation of a wireless sensor node against varying battery capacities and solar panel sizes. We compare our approach against [14], a state-of-the-art NAS methodology for EE-DNNs.

A. Neural Architecture Search (NAS)

1) Datasets and DNN Architecture: We selected four different vision datasets to evaluate our approach. All datasets are classification problems of small scale images (between 28×28 and 32×32 pixels). CIFAR10 [19] contains 10 different kinds of vehicles and animals, MNIST [20] contains handwritten digits between 0 and 9, FMNIST [21] contains 10 different classes of clothing, while Chars74k [22] contains all digits and letters of the English alphabet in upper and lower case, i.e. 64 classes in total. In all of our experiments, we use MobileNetV2 [23], a well known DNN architecture suitable for embedded deployment, as the backbone architecture from which our NAS approach samples different EE-DNNs.

2) Platform-specific Energy Modeling: For evaluating the consumed energy $e_{\text{DNN}}(c)$ of a DNN candidate when exiting at exit c according to Eq. (2), we performed a series of measurements of the energy consumption of the node during the inference of several differently scaled EE-DNNs obtained from our NAS optimization approach for CIFAR10. All measurements were performed on a NRF52840 SoC, which is a low-power microcontroller with a Cortex-M4F core. The targeted MCU has 256 kB RAM, 1 MB Flash (ROM), a FPU, and runs at a constant clock speed of 64 MHz. These measurements were then used to fit a linear model that estimates $e_{\text{DNN}}(c)$ from the number of MACs as

$$\hat{e}_{\text{DNN}}(c) = \alpha \cdot \#\text{MACs}(c) + \beta \tag{10}$$

where α is the slope of the line, β is the y-intercept and $\#\text{MACs}(c)$ is the number of MACs computed when taking the exit c. The model achieves an R-squared value of 0.75, which indicates a good fit. The obtained regression line is shown in Figure 3.

3) EENAS Validation: We evaluated the Inverted Generational Distance (IGD) and Hypervolume (HV) improvements obtained by our NAS approach over time for four different datasets, see Figure 4. For the computation of the IGD, we use the front of non-dominated points obtained by the optimization as the reference set. We performed this optimization for all four datasets for 50 samples and report the results in Figure 5, which plots the non-dominated points resulting from

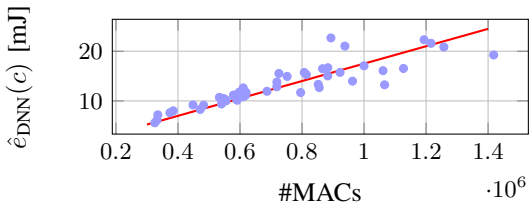


Fig. 3: Estimation of the energy consumption of a NRF52840 SoC when running an inference of a DNN until exit c in dependence of the number of MACs.

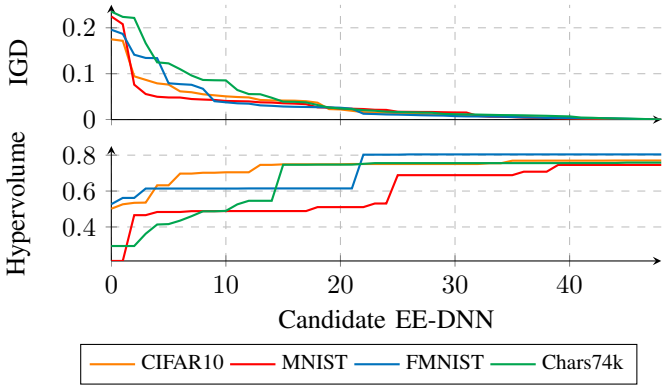


Fig. 4: HV (higher is better) and IGD (lower is better) improvements obtained by our multi-objective optimization approach for different datasets over 50 explored DNN candidates.

the optimization, i.e., the trade-offs in target space between the total accuracy Acc_{total} and the number of MACs.

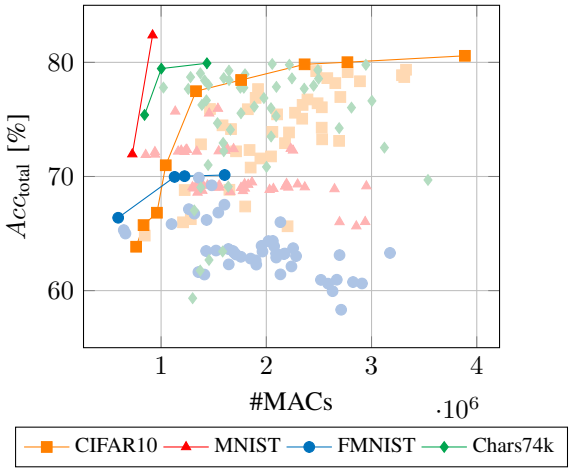


Fig. 5: Pareto fronts obtained by our approach for four different datasets (bold colored points). Light colored points denote dominated sampled DNNs explored by the optimizer. The accuracy Acc_{total} is maximized while the number of MACs is minimized.

In the plot, we highlight Pareto-optimal points in bold

colors, while we show dominated points in lighter colors. In all experiments shown in Figure 5, we set the trade-off parameter τ between the training accuracy Acc_{a} and the simulation accuracy Acc_{sim} in Eq. (7) to $\tau = 0.8$ when calculating Acc_{total}.

Both IGD and HV indicators show a monotonic improvement over the 50 samples used for the search. For all four datasets, the HV stabilizes towards the end of the optimization between 0.7 and 0.8, showing that the Pareto fronts resulting from the optimization cover the target space well. Looking at the Pareto optimal tradeoffs found for each dataset in Figure 5, the accuracy achieved by the models is at first glance slightly lower than expected when compared to the accuracy that can be achieved with state-of-the-art models [24]–[26]. However, there are two reasons for this: First, our NAS approach is optimized for a small MCU, which means that it imposes tight memory and energy constraints on the search. As a result, many architectural trade-offs that could achieve state-of-the-art accuracy are infeasible and therefore never explored by the optimizer because they would require too much memory or be to computational expensive for the target MCU. Second, and more importantly, the accuracy Acc_{total} shown in Figure 5 has been evaluated by a combination of the average accuracy of the EE-DNN over all exits, and the accuracy achieved under simulated constraints of an energy harvesting system. As a result, the simulated accuracy is typically slightly lower than the maximally achievable accuracy of the EE-DNN because it accounts for the fact that not always the ideal exit with the highest confidence and accuracy can be taken, but an earlier one due to limited energy availability. Therefore, the results paint a more realistic picture of how the EE-DNNs would perform under ENO on the respective datasets. With this in mind, we argue that our results clearly show that the explored EE-DNNs still achieve high accuracy despite limited energy.

4) Comparison with State-of-the-Art: To further emphasize the competitiveness of our approach, we compare our results with those obtained by a SotA method [14] that explores EE-DNNs for ENO when using the same datasets. In Table I, the results obtained by our methodology for different datasets are shown alongside the ones presented by the authors of [14]. For our method, we report the results obtained by the non-dominated candidate EE-DNN with the maximum accuracy Acc_{a} (first line) and the one with the minimum number of MACs (second line). For each EE-DNN, the column $\hat{\boldsymbol{a}}_{\mathbf{1}} \sim \hat{\boldsymbol{a}}_{\boldsymbol{C}}$ denotes the upper and lower bounds of the range of accuracies $\hat{a}_c$ obtained for all exits $c \in \{1, \ldots, C\}$.

We can see that our method achieves mostly similar accuracy values, while having a small memory footprint and a small number of MACs. The architectures obtained by our NAS approach manage to be competitive in terms of RAM usage. However, the authors of [14] do not report the ROM usage of their architectures. Besides, the FRAM non-volatile memory technology used in HarvNet is only available in specific low-power microcontrollers, which makes the architectures obtained by our method deployable on a broader range of devices. The memory footprint of the DNNs obtained by our method is larger than the one obtained by HarvNet, but it is still small enough to be run on a low-power microcontroller.

TABLE I: Comparison of the neural network architectures optimized by our method with ones obtained by HarvNet [14].

Dataset	**Method**	$\hat{\boldsymbol{a}}_{\mathbf{1}} \sim \hat{\boldsymbol{a}}_{\boldsymbol{C}}$ [%]	**ROM** [kB]	**RAM** [kB]	**#MACs** [k]
CIFAR-10	Proposed (max. Acc_{a})	79.0∼81.6	665	10	1033∼1945
	(min. #MACs)	42.4∼62.7	138	5	172∼382
	HarvNet	71.2∼77.6	n.d.	118	200∼347
MNIST	Proposed (max. Acc_{a})	98.7∼99.0	220	3	131∼460
	(min. #MACs)	98.1∼98.6	141	3	91∼366
	HarvNet	98.4∼98.6	n.d.	6	14∼14.1
FMNIST	Proposed (max. Acc_{a})	90.7∼91.1	211	6	388∼803
	(min. #MACs)	89.6∼89.8	154	2	128∼296
	HarvNet	83.9∼93.2	n.d.	29	30∼1114
Chars74K	Proposed (max. Acc_{a})	50.9∼66.3	167	8	407∼719
	(min. #MACs)	40.1∼62.0	260	4	143∼422
	HarvNet	70.9∼83.7	n.d.	21	59∼220

B. Energy Neutral Operation (ENO)

In this set of experiments, we evaluate the accuracy achieved by the DNNs obtained by different NAS methodologies when being driven by different energy management policies. We use a simulator written in Python to simulate the energy harvesting device and the energy management policy. Historical energy harvesting traces are obtained from the PVGIS platform[1], at the location (50° 42' 28.4544" N, 3° 50' 57.1776" E) on the 4th of May 2017.

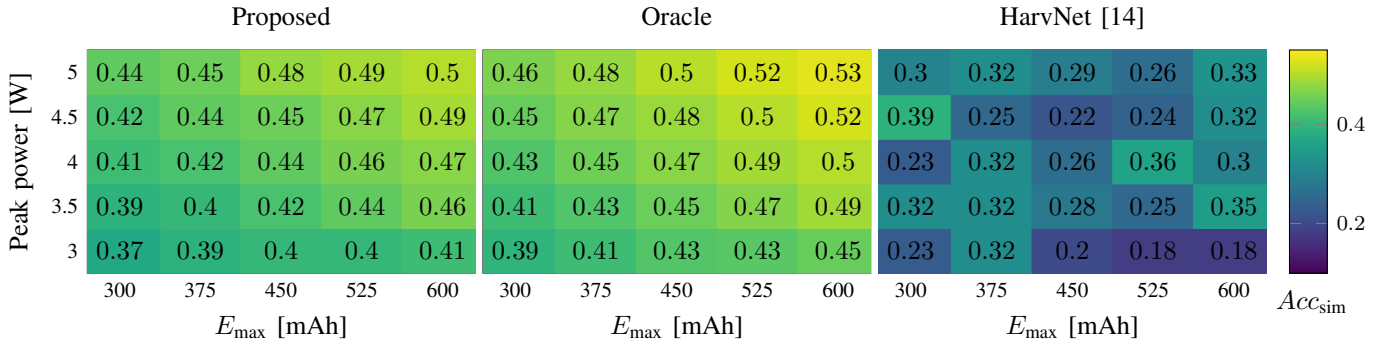
We evaluated the accuracy Acc_{sim} and the relative difference in remaining energy ΔE_{rem} at the end of the day

$$\Delta E_{\text{rem}} = \frac{E_{\text{rem}}(N) - E_{\text{rem}}(0)}{E_{\text{rem}}(0)}$$

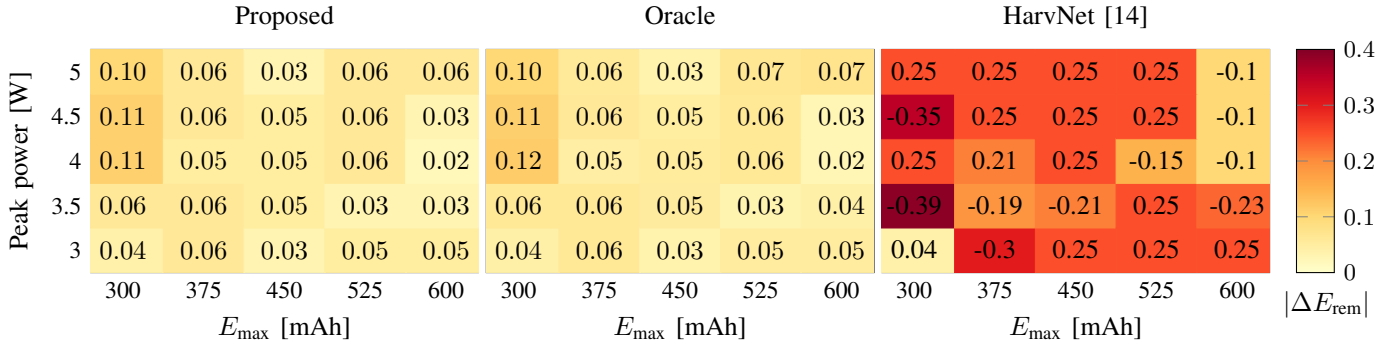
of an EE-DNN obtained by our EENAS over the FMNIST dataset while varying the peak wattage of the solar panel from 3W to 5W and the battery capacity E_{max} from 300mAh to 600mAh. We compare the obtained accuracies with the ones obtained by a SotA method [14] and an oracle agent who has a perfect knowledge of whether the output of an early exit is correct or not at the time of inference. We assume an accurate energy harvested prediction (i.e., $\bar{E}_{\text{harv}}(n) = E_{\text{harv}}(n)\ \forall n \in \{0, \ldots, N-1\}$) and we set $E_{\text{rem}}(0) = 0.5 \cdot E_{\text{max}}$, $N = 48$ and $K = 60$.

The results of the experiment, presented in Figure 6, show that our methodology achieve Acc_{sim} values close to the ones obtained by the oracle (with differences in accuracy ranging from 0.02 to 0.04), for each energy constraint, while performing strictly better than the related work (with differences in accuracy ranging from 0.07 to 0.36). Furthermore, these close-to-optimal accuracies are obtained while achieving ENO: the values of ΔE_{rem} are always greater than zero (i.e., the requirement specified in Eq. (5) is never violated) and there

[1] https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html



(a) Values of Acc_{sim}.



(b) Values of ΔE_{rem}.

Fig. 6: Achieved accuracy Acc_{sim} and ΔE_{rem} values for different solar panel's peak power and battery capacity $E_{\max}$ for our proposed approach, an oracle, and the HarvNet [14] on a real solar energy trace.

is, at worse, a surplus of energy of 11% of $E_{\text{rem}}(0)$ at the end of the day. Our methodology scales well as the energy requirements become more relaxed, being able to utilize more energy to achieve higher accuracy values.

On the contrary, the SotA method performs poorly when the energy constraints are unbalanced (i.e., high battery capacity and low peak power), reaching accuracies as low as 14%. It also often violates the ENO constraint (40% of the tested pairs of constraints), by as much as 39% of the initial energy. As the constraints get more relaxed, the method does not utilize all the energy available and often leaves a surplus of energy at the end of the day that is greater than the one obtained by both the Oracle and the proposed approach.

None of the energy management policies violate the minimal energy constraint specified in Eq. (4).

C. Energy Management Overhead

We evaluated the overhead of our energy management policy when deployed on a in terms of computational overhead (time) and energy of the three algorithms for energy neutral budget computation (Eq. (8)), determining a maximal exit schedule (Eq. (9)), and finally choosing an early-exit based on its entropy (Algorithm 1). We measured the cumulated execution time and energy overhead required to compute the energy budgets, for solving the UKPs shown in Eq. (9) and for the online policy shown in Algorithm 1 with $C = 6$, $K = 60$ and $N = 24$, for an entire day. We performed the measurements on the aforementioned NRF52840 SoC. The measurements, displayed in Table II, show that the overhead of our energy management policy is negligible compared to the rest of the workload of the sensor node: according to our energy model, the energy cost of a single inference of an EE-DNN of 1.2M MACs is around 20 mJ, which is about the same amount of energy than the overhead of the energy management.

TABLE II: Measured overheads of the three energy management policy tasks cumulated over a full day (24h).

Measurement	**Time** [ms]	**Energy** [mJ]
Energy-neutral budget computation	78.3	4.3
Maximal Early-Exit scheduling	302.4	16.6
Entropy-Aware Online Policy	2	0.11

V. Conclusion

In this work, we presented a novel multi-objective exploration and optimization methodology for EE-DNNs under given energy and device resource constraints in energy harvesting IoT systems. Furthermore, we contributed a strategy for runtime energy management (EM) by selecting the best early exit of the EE-DNN for each inference while ensuring ENO. Our results on four machine learning vision datasets

show that our approach can explore feasible EE-DNNs that achieve high accuracy while still allowing ENO. Furthermore, our results show that we outperform related work in accuracy when driven by our energy management policy in a simulated energy harvesting system, while having similar memory and computational requirements.

REFERENCES

[1] M. R. M. Kassim and A. N. Harun, "Applications of WSN in agricultural environment monitoring systems," in *2016 ICTC*, 2016, pp. 344–349.

[2] J. Wägele, P. Bodesheim, S. J. Bourlat, J. Denzler, M. Diepenbroek, V. Fonseca, K.-H. Frommolt, M. F. Geiger, B. Gemeinholzer, F. O. Glöckner, T. Haucke, A. Kirse, A. Kölpin, I. Kostadinov, H. S. Kühl, F. Kurth, M. Lasseck, S. Liedke, F. Losch, S. Müller, N. Petrovskaya, K. Piotrowski, B. Radig, C. Scherber, L. Schoppmann, J. Schulz, V. Steinhage, G. F. Tschan, W. Vautz, D. Velotto, M. Weigend, and S. Wildermann, "Towards a multisensor station for automated biodiversity monitoring," *Basic and Applied Ecology*, vol. 59, pp. 105–138, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1439179122000032

[3] P. Sixdenier, S. Wildermann, and J. Teich, "GRES: Guaranteed Remaining Energy Scheduling of Energy-harvesting Sensors by Quality Adaptation," in *13th Mediterranean Conference on Embedded Computing, MECO 2024, Budva, Montenegro, June 11-14, 2024*. IEEE, 2024, pp. 1–5. [Online]. Available: https://doi.org/10.1109/MECO62516.2024.10577838

[4] M. Deutel, F. Hannig, C. Mutschler, and J. Teich, "On-device training of fully quantized deep neural networks on Cortex-M microcontrollers," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 44, no. 4, pp. 1250–1261, 2025. [Online]. Available: https://doi.org/10.1109/TCAD.2024.3484354

[5] M. Sabih, B. Sesli, F. Hannig, and J. Teich, "Accelerating DNNs using weight clustering on RISC-V custom functional units," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2024, Valencia, Spain, March 25-27, 2024*. IEEE, 2024, pp. 1–2. [Online]. Available: https://doi.org/10.23919/DATE58400.2024.10546844

[6] M. Sabih, M. Yayla, F. Hannig, J. Teich, and J. Chen, "Robust and tiny binary neural networks using gradient-based explainability methods," in *Proceedings of the 3rd Workshop on Machine Learning and Systems, EuroMLSys 2023, Rome, Italy, 8 May 2023*, E. Yoneki and L. Nardi, Eds. ACM, 2023, pp. 87–93. [Online]. Available: https://doi.org/10.1145/3578356.3592595

[7] M. Deutel, G. Kontes, C. Mutschler, and J. Teich, "Combining multi-objective bayesian optimization with reinforcement learning for TinyML," *ACM Trans. Evol. Learn. Optim.*, Jan. 2025, just Accepted. [Online]. Available: https://doi.org/10.1145/3715012

[8] C. Heidorn, M. Sabih, N. Meyerhöfer, C. Schinabeck, J. Teich, and F. Hannig, "Hardware-aware evolutionary explainable filter pruning for convolutional neural networks," *Int. J. Parallel Program.*, vol. 52, no. 1-2, pp. 40–58, 2024. [Online]. Available: https://doi.org/10.1007/s10766-024-00760-5

[9] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[10] S. Scardapane, D. Comminiello, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Differentiable branching in deep networks for fast inference," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4167–4171.

[11] X. Dai, X. Kong, and T. Guo, "EPNet: Learning to exit with flexible multi-branch network," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 235–244.

[12] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations*, 2018.

[13] M. Phuong and C. H. Lampert, "Distillation-based training for multi-exit architectures," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1355–1364.

[14] S. Jeon, Y. Choi, Y. Cho, and H. Cha, "HarvNet: Resource-optimized operation of multi-exit deep neural networks on energy harvesting devices," in *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. Helsinki Finland: ACM, Jun. 2023, pp. 42–55.

[15] S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv preprint arXiv:2304.11127*, 2023.

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[17] M. Deutel, P. Woller, C. Mutschler, and J. Teich, "Energy-efficient deployment of deep learning applications on Cortex-M based microcontrollers using deep compression," in *MBMV 2023; 26th Workshop*. VDE, 2023, pp. 1–12.

[18] V. Poirriez, N. Yanev, and R. Andonov, "A hybrid algorithm for the unbounded knapsack problem," *Discrete Optimization*, vol. 6, no. 1, pp. 110–124, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528608000686

[19] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[22] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *International conference on computer vision theory and applications*, vol. 1, 2009, pp. 273–280.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[24] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*. Pmlr, 2015, pp. 562–570.

[25] E. Harris, A. Marcu, M. Painter, M. Niranjan, A. Prügel-Bennett, and J. Hare, "FMix: Enhancing mixed sample data augmentation," *arXiv preprint arXiv:2002.12047*, 2020.

[26] A. J. Newell and L. D. Griffin, "Multiscale histogram of oriented gradient descriptors for robust character recognition," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 1085–1089.