Early-Exit Forecasting of Deep Neural Networks on Energy-Harvesting Edge Devices

Pierre-Louis Sixdenier(✉), Mark Deutel, Stefan Wildermann, and Jürgen Teich

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany
{pierre-louis.e.sixdenier,mark.deutel,stefan.wildermann,juergen.teich}@fau.de

Abstract. Ambient energy harvesting (EH) allows to extend the operational lifetime of battery-constrained wireless sensor nodes (WSNs) deployed in remote environments. The deployment of deep neural networks (DNNs) on EH-WSNs is of great interest for a wide range of applications, including environmental monitoring and wildlife tracking. However, sustaining continuous DNN inference on such systems is infeasible under harsh energy constraints imposed by limited battery capacity and varying amounts of energy harvested. Early-exit deep neural networks (EE-DNNs) allow to dynamically adapt the energy requirements of a DNN inference by integrating multiple exit points within the DNN architecture, enabling shorter inference paths for samples that can be classified with high confidence. This paper presents a novel policy that leverages a secondary small neural network called forecaster, which is trained to predict the most efficient exit point to be used by the EE-DNN for a given sample. This reduces the computational overhead of evaluating multiple early exits sequentially, as is the case with a default confidence-based policy. Empirical evaluation on two vision datasets demonstrates that our policy achieves a up to 23% higher accuracy compared to a default entropy-based policy. We also show that using a forecaster results in a higher accuracy under energy constraints than related work, while ensuring energy-neutral operation of the system.

Keywords: Energy harvesting · Early-exit neural networks · Edge devices.

1 Introduction

Energy harvesting (EH) may help to extend the operational lifetime of battery-powered Wireless Sensor Networks (WSNs) deployed in remote settings such as agricultural monitoring [8]. However, the intermittent nature of harvested energy makes continuous operation challenging. Energy management schemes have been proposed to maximize different objectives, e.g., duty cycle or task-level accuracy, while enforcing state-of-charge constraints to achieve energy-neutral operation (ENO) [14].

Local Deep Neural Network (DNN) inference on sensor nodes allows to early process data and thus reduce network overhead and can improve responsiveness of an embedded system. But it is constrained by limited memory and compute

resources. Hence, DNN compression and architecture optimization techniques like pruning, quantization, weight clustering, and neural architecture search (NAS) have proven essential to fit models to embedded systems [3, 5, 12].

Early-exit DNNs (EE-DNNs) [15] provide multiple classification exits. By choosing the inference result of an early exit, it is possible to trade off reduced computation and energy for accuracy. In energy-harvesting WSNs (EH-WSNs), the computational flexibility of an EE-DNN can be leveraged by an energy manager to select early exits that ensure ENO while maximizing accuracy.

In related approaches, exit selection can be based on confidence thresholds [13, 15] or reinforcement learning (RL) [6], which are not optimal for the dynamic energy availability in EH-WSNs. In particular, [6, 15] greedily iterate through exits until a confidence threshold is reached can cause significant computational overhead, while RL-based methods such as [6] can be computationally intensive and require retraining after deployement to adapt to the rapidly changing energy conditions in EH-WSNs.

In this paper, our contributions are twofold: (1) Presented is a *forecaster* DNN that predicts the exit with the highest predicted accuracy for each inference of an EE-DNN (Section 3.1) and (2), a *runtime energy-management strategy* that selects per-inference exits while ensuring ENO (Section 3.2). We validate our approach on two vision datasets and report accuracy across varying battery capacities and photovoltaic peak powers (Section 4).

2 Fundamentals

This section presents the fundamentals of Early-Exit DNNs, the system model of EH-WSNs and the problem of energy-neutral operation (ENO).

2.1 Early-Exit DNNs

EE-DNNs [15] denote a class of DNNs that are used to speed up inference of DNNs by adding additional side classifiers (sometimes also just called early exits) to a backbone DNN, see Figure 1a for a schematic overview. Thereby, EE-DNNs allow an inference to exit at an early stage of its feature extraction pipeline for "simple" input samples, if the DNN is highly confident about the classification. Assuming that input samples are predominantly "simple", using EE-DNNs reduces the overall inference time and the computational overhead while still allowing dynamic adjustment and allocation of the full feature extraction pipeline for complex edge-case samples.

In a "standard" EE-DNN, i.e., [15], the policy of whether an inference should terminate early for a given input sample is to greedily evaluate the entropy of each early exit sequentially, until either the last exit or an early exit with a low entropy (i.e., high predictability) is reached. The entropy $\lambda(\hat{y}_c)$ of an early exit $c \in \{1, \ldots, C\}$, with C being the total number of exits, is computed using Shannon's entropy of the class probability vector $\hat{y}_c$ over all class labels $l \in \{0, \ldots, L-1\}$, see Eq. (1). Note that in Eq. (1), the entropy is additionally

divided by $\log(L)$, i.e., the maximum entropy for all L classes. This means that $0 \leq \lambda(\hat{y}_c) \leq 1$, with $\lambda(\hat{y}_c) = 0$ denoting maximum confidence in a single class, and $\lambda(\hat{y}_c) = 1$ denoting that every class is predicted with equal probability.

$$\lambda(\hat{y}_c) = -\frac{1}{\log(L)} \sum_{l=0}^{L-1} \hat{y}_{c,l} \cdot \log\left(\hat{y}_{c,l}\right) \tag{1}$$

Due to their ability to provide adaptable quality of service based not only on internal confidence and predictability, but also on external factors such as resource and energy availability, EE-DNNs have been proposed in literature for use on energy-harvesting embedded systems [6, 13]. However, a drawback that all of these approaches have in common is that they rely on the greedy entropy evaluation policy of their EE-DNNs during inference. This means that in a worst-case scenario, the EE-DNN must be evaluated up to its final exit including all intermediate exits, resulting in greater computational overhead than if only the backbone network were executed to the final exit. To mitigate this problem, we propose an alternative evaluation policy in this paper that uses a forecaster network to predict for each sample which early exit to take based on an early feature map (see Section 3.1). This means that the overhead of evaluating multiple early exits sequentially is replaced with only having to calculate a single, small forecaster network plus the selected exit instead.

2.2 System Model and Energy-Neutral Operation

We consider a battery-powered sensor node equipped with a photovoltaic harvester. The node operates over a day divided into N discrete time slots indexed by $n \in \{0, \ldots, N-1\}$. During a time slot n, the harvester supplies an amount of energy $E_{\text{harv}}(n)$, which is stored in a rechargeable battery of finite capacity $E_{\max}$.

The node periodically acquires sensor data and executes inference tasks at a period T. We denote by $K = 24h/N/T$ the amount of inferences performed within a time slot and by $k \in \{0, \ldots, K-1\}$ the index of the k-th inference in a slot. The employed EE-DNN exposes C internal exits, executing the network up to exit $c \in \{1, \ldots, C\}$, i.e., including (parts of) the backbone and, in case of the proposed approach, the forecaster, consumes an energy of $e_{\text{DNN}}(c)$. Let $c = 0$ denote skipping the inference (no DNN execution), which has an energy cost $e_{\text{DNN}}(0) = 0$.

Let the binary decision variable $X_{n,k,c} \in \{0, 1\}$ indicate whether the k-th inference in slot n should terminate at exit c ($X_{n,k,c} = 1$) or not ($X_{n,k,c} = 0$). Moreover, exactly one exit is to be selected per inference (see respective constraint in Eq. (6)). The energy consumed by the node inside a time slot n is then denoted by

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

The state of charge of the battery evolves according to the balance between harvested and consumed energy. Denoting by $E_{\text{rem}}(n)$ the state of charge at the

beginning of a slot n, the residual energy at the beginning of the next slot is

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

where $\lceil x \rceil^{+} = \max\{x, 0\}$ ensures non-negativity.

Energy-neutral operation (ENO) is the fundamental operating principle for EH-WSNs, as it ensures that the network can operate perpetually without exhausting the energy of any node, maximizing the availability of the network [7]. In literature [2, 7], for analytical feasibility, ENO is often defined over a tractable timescale, such as a day. This is especially relevant for the solar energy harvesting systems that we focus on in this work, where the energy harvesting profile shows a pseudo-periodic behavior over a 24-hour cycle. Accordingly, ENO can be formulated by the following constraints:

1. The state of charge must never fall below a safety threshold $E_{\min}$,

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

2. The state of charge reached at the end of the day must not be lower than the one that was available at the beginning of the day,

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

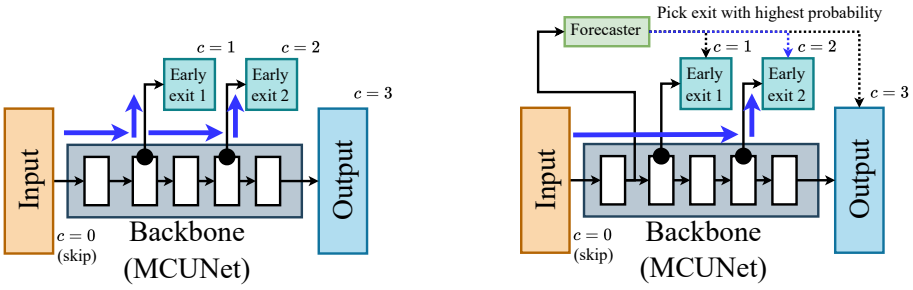
For each possible exit c, the EE-DNN outputs a class probability vector $\hat{Y}_{n,k,c}$ for the k-th inference of slot n, which is used to predict the ground-truth label $Y_{n,k,c}$. Selecting exits for all inferences so as to maximize the number of correct predictions over the day, while satisfying the ENO constraints, can be expressed as the following integer linear program (ILP):

$$\begin{aligned} \max_{X_{n,k,c}} \; & Acc_{\text{sim}} = \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} \sum_{c=0}^{C} \left[\!\left[\hat{Y}_{n,k,c} = Y_{n,k,c}\right]\!\right] \cdot X_{n,k,c} \\ \text{s.t.} \; & \sum_{c=0}^{C} X_{n,k,c} = 1 \\ & \forall n \in \{0, \ldots, N-1\} \\ & \forall k \in \{0, \ldots, K-1\} \\ & \text{Eqs. (2), (3), (4), (5),} \end{aligned} \tag{6}$$

Solving the ILP above requires knowledge of the true labels $Y_{n,k,c}$ and the future energy arrivals $E_{\text{harv}}(n)$, which are not available at runtime. To address these limitations, we adopt a two-stage, best-effort approach that we describe in the following.

3 Methodology

In [10], the use of a "low-cost prediction engine" instead of greedy evaluation was proposed to improve the overall energy efficiency of EE-DNNs. In this work,



(a) Greedy policy.

(b) Proposed forecaster policy.

Fig. 1. Schematic comparison of the greedy execution policy from [15] (Figure 1a) versus our proposed forecaster policy (Figure 1b). The blue arrows denote the paths taken by both policies under the assumption that the second early exit is eventually taken. Whereas, with the greedy policy, every intermediate exit must be executed sequentially, intermediate early exits do not need to be evaluated using the forecaster.

we adapt this forecasting method (cf. Section 3.1) and combine it with a runtime energy management policy (cf. Section 3.2) to enable adaptable quality of service for DNN execution on EH-WSNs with minimal computational overhead. Thereby, our approach not only ensures ENO but also maximizes accuracy given tight energy budget constraints.

3.1 Exit Forecasting

We show our proposed forecasting evaluation policy in comparison to the greedy entropy policy in Figure 1. The blue arrows in both Figure 1a and Figure 1b show the paths in blue taken when executing the EE-DNN for a given input sample assuming that early exit $c = 2$ is eventually selected by both policies. As can be seen, the advantage of using a forecasting network is that it provides a predictable, static overhead for determining which early exit to take, rather than requiring sequential evaluation of a varying number of early exits. This means that, especially in the worst-case scenario of evaluating the entire EE-DNN up to the final exit, the use of the forecaster may reduce inference latency and energy consumption as our experimental results will witness.

For the forecaster, we choose a small trainable neural network which takes an intermediate feature map from the backbone as input and outputs a probability vector over all $c \in \{1, \ldots, C\}$ possible exits (excluding $c = 0$ which we define to skip execution of the EE-DNN completely). The exit with the highest probability is then selected for direct execution without any branching into other intermediate side classifiers.

We propose training an EE-DNN with its forecaster as a three-step process:

1. The EE-DNN is trained with all its side classifiers using a joint training strategy. In joint training, a single loss $\mathcal{L}$ is calculated by combining the

cross-entropy $\mathcal{H}$ between the ground truth y from the training dataset and the predicted distribution $\hat{y}_c$ of each early exit $c \in \{1, \ldots, C\}$ as a weighted sum, see Eq. (7).

$$\mathcal{L}(y, \hat{y}) = \sum_{c=1}^{C} w_c \, \mathcal{H}(y, \hat{y}_c) \tag{7}$$

The weights $w_c \in [0, 1]$ used for summing the losses of each exit are tunable hyperparameters. In our experiments, choosing smaller weights for earlier exits and larger weights for later ones yielded the best results.

2. The parameters of the trained EE-DNN are frozen to train the forecaster.
3. Inferences of the frozen EE-DNN are performed for every sample in the training set. This is done to (a) get the intermediate feature maps that will be used as inputs for the forecaster and (b) find the earliest exit for each sample that correctly predicts the class according to the ground truth y of the training dataset and that has an entropy $\lambda(\hat{y}_c)$ below a predefined threshold th^{up}. The index of the earliest exit found is then used as label for training the forecaster. In our experiments, to maximize the number of samples with varying complexity and to stabilize the forecaster's convergence during training, we additionally employed significant data augmentation [4] during forecaster training to generate numerous invariants of varying complexity from a single training sample.

3.2 Energy Management

Over the day, the Energy Manager (EM) performs two tasks at different granularities of time steps:

- *Energy-Neutral Budget Computation*: At the beginning of each day of operation, a dynamic programming algorithm computes the maximal energy budget $E_{\text{b}}(n)$ for every time slot n that ensures ENO. The corresponding optimization problem is stated in Eq. (8).
- *Forecast-Aware Online Policy*: For each inference k, a runtime policy that, given $E_{\text{b}}(n)$, selects the exit to chose for each inference n, k based on the prediction made by the forecasting network but ensuring that the energy budget $E_{\text{b}}(n)$ is never exceeded. We introduce this policy in Algorithm 1.

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

Energy-Neutral Budget Computation This task is performed once at the beginning of each day to compute an energy-neutral budget $E_{\text{b}}(n)$ for each time slot $n \in \{0, \ldots, N-1\}$. It takes as an input a trace of predicted harvested energy $\bar{E}_{\text{harv}}(n)$ which is obtained by applying an Exponentially Weighted Moving Average (EWMA) filter to historical measurements. This is a common state-of-the-art approach, see, e.g., [7]. An allocation of energy-neutral energy budgets

Algorithm 1 Forecast-Aware Online Policy

1: **Input:** n, k, $E_\mathrm{b}(n)$, $c_\mathrm{f}(n,k)$
2: **Output:** $X_{n,k,c}$
3: **if** $e_\mathrm{DNN}(1) \leq E_\mathrm{b}(n)$ **then** ▷ If at least one exit can be scheduled
4: $\quad c_\mathrm{max}(n,k) \leftarrow \mathrm{argmax}_c \{e_\mathrm{DNN}(c), c \in \{1,\ldots,C\} \mid e_\mathrm{DNN}(c) \leq E_\mathrm{b}(n)\}$
5: $\quad c \leftarrow min\{c_\mathrm{max}(n,k), c_\mathrm{f}(n,k)\}$
6: $\quad$ Run inference for exit c on input n,k
7: $\quad X_{n,k,c} \leftarrow 1$
8: $\quad E_b(n) \leftarrow E_b(n) - e_\mathrm{DNN}(c)$ ▷ Update the remaining energy budget
9: **end if**

$E_\mathrm{b}(n)$ for each time slot n can be obtained by solving the following optimization problem (adapted from [2]) with a dynamic programming algorithm:

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

When the prediction $\bar{E}_\mathrm{harv}(n)$ is conservative (i.e., $\bar{E}_\mathrm{harv}(n) < E_\mathrm{harv}(n)\ \forall n$) and the energy dynamics updates follow Eqs. (2) and (3), a solution to Eq. (8) ensures ENO over the full day.

Forecast-Aware Online Policy Algorithm 1 describes our policy for selecting the exit branch of the EE-DNN to use for inference k of time slot n. It takes as input the energy budget $E_\mathrm{b}(n)$ computed at the start of the day and the exit forecast $c_\mathrm{f}(n,k)$ computed by the forecaster. The policy first checks whether at least one DNN exit can be scheduled given the current budget (Line 3). If so, it computes the maximal allowed exit $c_\mathrm{max}(n,k)$ that fits the budget (Line 4). Then, the policy selects the exit c that is the earliest between the forecasted exit $c_\mathrm{f}(n,k)$ and the maximal allowed exit $c_\mathrm{max}(n,k)$ (Line 5). The selected exit is executed and recorded (Line 7), and the remaining energy budget is decremented accordingly (Line 8).

4 Experiments

We evaluate our proposed methodology on the CIFAR-10 [9] and the Chars74K [1] datasets. In our experiments, we use MCUNet [11] (mcunet-in4), a state-of-the-art architecture for MCU-based classification applications, as the base architecture of our EE-DNNs. We attach two early exits to the base architecture, one after the sixth inverted residual block and the other after the ninth inverted residual block,

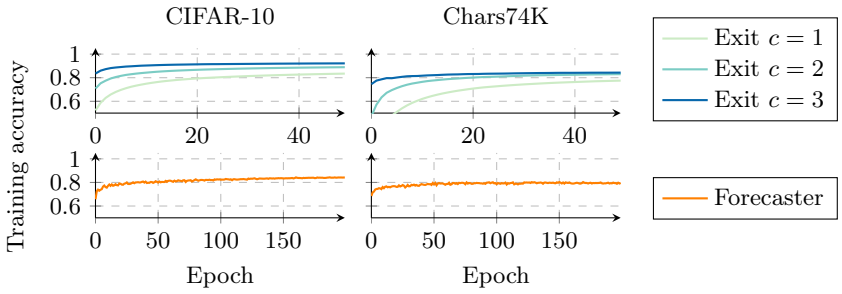


Fig. 2. Training accuracy of the EE-DNN on the CIFAR-10 and Chars74K datasets. The lines in the first row correspond to the validation accuracy obtained for each exit during training. In the second row, the lines denote the forecaster's accuracy in predicting the most efficient exit for each sample during training.

resulting in a total of $C = 3$ exits. The two early exits $c = 1$ and $c = 2$ consist of an adaptive average pooling layer followed by a linear layer, while $c = 3$ is the original classifier of the MCUNet architecture. We set $th^{\text{up}} = 0.1$. The thresholds for the entropy-based policies are set to $th_0 = 0.01, th_1 = 1.96,$ and $th_2 = 1.85$, optimized following the procedure described in [13].

4.1 Training and Evaluation

For both CIFAR-10 and Chars74K, we train the considered EE-DNN for 50 epochs with a batch size of 256 and Adam as the optimizer (with learning rate 0.001 and weight decay $1e^{-5}$). We show the class accuracy of the validation dataset of each of the three early exits during training separately in the first row of Figure 2, (i.e., without considering the entropy or forecaster policy). For both datasets, the last exit $c = 3$ achieved state-of-the art accuracy with the two early exits $c = 1$ and $c = 2$ ending up as faster-to-compute trade-offs at a lower accuracy.

In the second row of Figure 2, we present the accuracy with which the forecaster models predict the most efficient early exit (i.e. the earliest exit that yields the correct class and is below the entropy threshold as defined in Section 3.1) during their training on the completely trained EE-DNNs for every sample in the CIFAR-10 and Chars74K validation datasets. As the forecaster model, we use two small inverted residual blocks (both with 16 output channels) followed by dropout, adaptive average pooling and a linear layer as the classifier. We trained both forecasters for 200 epochs, using Adam as the optimizer with cosine annealing as a learning rate scheduler.

In Table 1, we show the confusion matrices of the two forecasters predicting the most efficient early exit for each sample in the validation datasets of CIFAR-10 and Chars74K after their training. It can be seen that for both datasets the forecasters predominantly predicted the most efficient exit as defined in Section 3.1, i.e., the diagonal axes, for a majority of the validation samples. As a result, when using the forecaster-based selection method, we observed an

Table 1. Confusion matrices of the forecasters trained for CIFAR-10 and Chars74K.

		Predicted		
		Exit 1	Exit 2	Exit 3
Label	Exit 1	2934	170	32
	Exit 2	235	1964	621
	Exit 3	36	485	3523

(a) CIFAR-10

		Predicted		
		Exit 1	Exit 2	Exit 3
Label	Exit 1	917	77	13
	Exit 2	119	274	87
	Exit 3	41	174	795

(b) Chars74K

accuracy of 0.89 for CIFAR-10 and 0.84 for Chars74K which in both cases is very close ($< 3\%$) to the accuracy achieved when always picking the last exit (i.e., the highest possible accuracy). In comparison, when using the entropy-based selection method with thresholds, we only observed an accuracy of 0.87 for CIFAR-10 and 0.78 for Chars74K. This clearly demonstrates the effectiveness of the forecaster method to correctly and robustly predict the exit that yields the best trade-off between efficiency and accuracy in comparison to the entropy-based selection method which, based on how its threshold is configured, might decide too greedy and exit DNN execution too early.

4.2 Energy Neutrality and Accuracy Results

In the following experiment, we assess the performance of DNNs produced by distinct NAS procedures when controlled by alternative energy management policies. A Python-based simulator was used to model both the energy-harvesting system and the considered policies. Historical solar irradiance traces were sourced from the PVGIS platform[1] for the coordinates 50°42'28.4544" N, 3°50'57.1776" E on all days between 4th May 2017 and 7th May 2017.

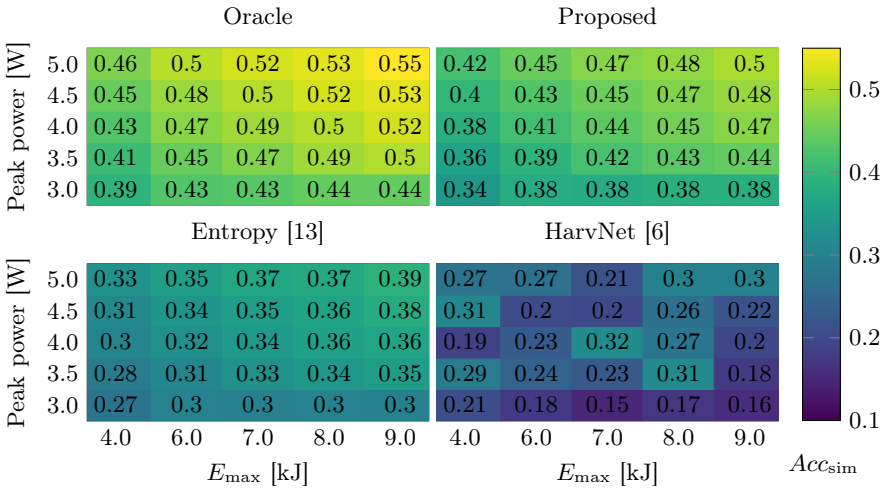
We report the classification accuracy while under energy constraints achieved in average over all days, denoted by Acc_{sim}, and the relative change in remaining battery energy at the end of the day,

$$\Delta E_{\text{rem}} = \frac{E_{\text{rem}}(N) - E_{\text{rem}}(0)}{E_{\text{rem}}(0)} \ , \tag{9}$$

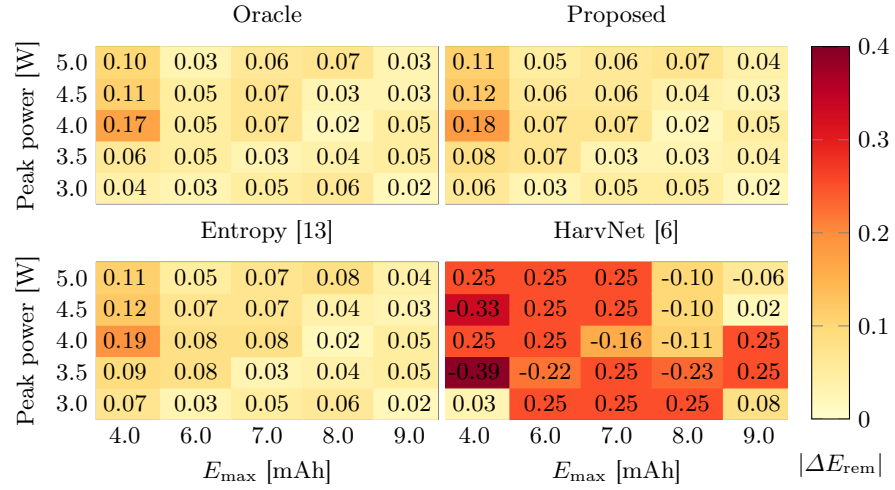
for the EE-DNN we designed for the CIFAR-10 dataset. The experiment iterates on different values of the solar panel peak power from 3 W to 5 W and the battery capacity $E_{\max}$ from 4.0 kJ to 9.0 kJ. Results are compared against a state-of-the-art RL-based baseline [6], an entropy-based policy [13], and an oracle that has perfect knowledge of early exits' correctness at inference time. We assume perfect energy-harvesting predictions (i.e., $\bar{E}_{\text{harv}}(n) = E_{\text{harv}}(n)\ \forall n \in \{0, \ldots, N-1\}$) and set $E_{\text{rem}}(0) = 0.5\, E_{\max}$, $N = 48$, and $K = 60$.

The obtained results can be found in Figure 3. As can be seen, our approach obtains Acc_{sim} values that are close to the ones obtained by the oracle (with a difference of 6% of accuracy at maximum) across the tested energy configurations, and consistently outperforms the state-of-the-art baselines with gains ranging

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



(a) Values of Acc_{sim}.



(b) Values of ΔE_{rem}.

Fig. 3. Achieved accuracy Acc_{sim} and ΔE_{rem} values for different solar panel's peak powers (y-axis) and battery capacities $E_{\max}$ (x-axis) for our proposed forecaster approach, an oracle, an entropy-based method [13], and HarvNet [6] on a solar energy trace.

from 7% to 23% of accuracy. Noticeably, these near-optimal accuracies are achieved while satisfying the energy-neutrality constraints: $\Delta E_{\text{rem}} \geq 0$ for all evaluated pairs of constraints (with a maximum surplus of about 18% of $E_{\text{rem}}(0)$).

Both the oracle, the entropy-based method, and our proposed approach always satisfy the ENO requirement, with ΔE_{rem} values that are close to zero when constraints are tight and that increase as constraints are relaxed, indicating that the energy budget is fully utilized when possible. By contrast, HarvNet often

fails to fully utilize available energy, leaving large end-of-day energy surpluses. It exhibits degraded performance under unbalanced energy constraints (e.g., large battery capacity combined with low peak power), with accuracies dropping to 15% in the worst cases, and violates the ENO requirement for 9 out of 25 of the tested configurations (up to a 39% shortfall relative to initial energy).

4.3 Hardware Implementation and Overhead Analysis

We evaluated the computational overhead of the forecaster DNN as well as the energy management policy introduced in Section 3.2 when deployed on a XIAO BLE[2] embedding a nRF52840 SoC. We measured, over a full day, the cumulative energy overhead of energy-budget computation and of the online policy implemented according to Algorithm 1, using $C = 3$, $K = 60$, and $N = 24$. We modeled the energy consumption of the forecaster DNN and of the EE-DNN for each exit using the number of multiply-accumulate operations (MACs) and the energy per MAC of the target platform, which we measured to be 0.175 µJ/MAC for the XIAO BLE.

The measurements, displayed in Table 2, show that the overhead of our energy management policy is negligible compared to the rest of the workload of the sensor node. Furthermore, the forecaster DNN inference overhead is significantly lower than the one of the EE-DNN inference, with a 185x reduction in energy consumption compared to the final exit of the EE-DNN.

Table 2. Measured overheads of the two energy management policy tasks cumulated over a full day (24h) and of the forecaster DNN and the EE-DNN.

Measurement	**Energy** [mJ]	**#MACs**
Over a full day (24h)		
Energy-Neutral Budget Computation	4.1	n/a
Forecast-Aware Online Policy	0.11	n/a
Per inference (n, k)		
Forecaster DNN Inference	9.13	0.5M
EE-DNN Inference		
Exit $c = 1$	185.6	10.6M
Exit $c = 2$	319.2	17.7M
Exit $c = 3$	1667.2	95M

5 Conclusion and Outlook

In this paper, we have presented a novel policy for deciding which exit point to use for each sample in an EE-DNN deployed on EH-WSNs. Our policy leverages a side neural network called forecaster, which is trained to predict the exit point

[2] `https://wiki.seeedstudio.com/XIAO_BLE/`

that will be used by the EE-DNN for a given sample. We demonstrate that our policy achieves a higher accuracy under energy constraints than related work, while ensuring energy-neutral operation of the system.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. de Campos, T.E., Babu, B.R., Varma, M.: Character recognition in natural images. In: International conference on computer vision theory and applications. vol. 1, pp. 273–280 (2009)
2. Caruso, A., Chessa, S., Escolar, S., Rincon, F., Lopez, J.C.: Task scheduling stabilization for solar energy harvesting internet of things devices. pp. 1–6 (2022)
3. Deutel, M., Kontes, G., Mutschler, C., Teich, J.: Combining multi-objective bayesian optimization with reinforcement learning for tinyml. ACM Transactions on Evolutionary Learning **5**(3), 1–21 (2025)
4. Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., Hare, J.: FMix: Enhancing mixed sample data augmentation. arXiv:2002.12047 (2020)
5. Heidorn, C., Sabih, M., Meyerhöfer, N., Schinabeck, C., Teich, J., Hannig, F.: Hardware-aware evolutionary explainable filter pruning for convolutional neural networks. Int. J. Parallel Program. **52**(1-2), 40–58 (2024)
6. Jeon, S., Choi, Y., Cho, Y., Cha, H.: 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. pp. 42–55. Helsinki Finland (2023)
7. Kansal, A., Hsu, J., Zahedi, S., Srivastava, M.B.: Power management in energy harvesting sensor networks. ACM TECS **6**(4), 32 (2007)
8. Kassim, M.R.M., Harun, A.N.: Applications of WSN in agricultural environment monitoring systems. In: 2016 ICTC. pp. 344–349 (2016)
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
10. Li, X., Lou, C., Chen, Y., Zhu, Z., Shen, Y., Ma, Y., Zou, A.: Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 8657–8665 (2023)
11. Lin, J., Chen, W.M., Lin, Y., cohn, j., Gan, C., Han, S.: MCUNet: Tiny deep learning on IoT devices. In: Advances in Neural Information Processing Systems. vol. 33, pp. 11711–11722. Curran Associates, Inc. (2020)
12. Sabih, M., Sesli, B., Hannig, F., Teich, J.: Accelerating DNNs using weight clustering on RISC-V custom functional units. In: DATE 2024, Spain, March 25-27, 2024
13. Sixdenier, P.L., Deutel, M., Teich, J.: Early-exit neural architecture search for energy-harvesting edge computing. In: 2025 IEEE 18th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). pp. 372–379 (2025)
14. Sixdenier, P.L., Wildermann, S., Teich, J.: GRES: Guaranteed remaining energy scheduling of energy-harvesting sensors by quality adaptation. In: Proceedings of the 13th Mediterranean Conference on Embedded Computing (MECO) (2024)
15. Teerapittayanon, S., McDanel, B., Kung, H.T.: Branchynet: Fast inference via early exiting from deep neural networks. In: 2016 23rd international conference on pattern recognition (ICPR). pp. 2464–2469. IEEE (2016)