2025 Design, Automation & Test in Europe Conference (DATE 2025)

SolarML: Optimizing Sensing and Inference for Solar-Powered TinyML Platforms

Hao Liu, Qing Wang, and Marco Zuniga
Delft University of Technology, Netherlands
{h.liu-8, qing.wang, m.a.zunigazamalloa}@tudelft.nl

***Abstract*—Machine learning models can now run on microcontrollers. Thanks to the advances in neural architectural search, we can automatically identify tiny machine learning (tinyML) models that satisfy stringent memory and energy requirements. However, existing methods often overlook the energy used during event detection and data gathering. This is critical for devices powered by renewable energy sources like solar power, where energy efficiency is paramount. To address it, we introduce *SolarML*, a solution designed specifically for solar-powered tinyML platforms, which optimizes the end-to-end system's inference accuracy and energy consumption, from data gathering and processing to model inference. Considering two applications of gesture recognition and keywords spotting, SolarML has the following contributions: 1) a hardware platform with an optimal event detection mechanism that reduces event detection costs by up to 10× compared to state-of-the-art alternatives; 2) a joint optimization framework *e*NAS that reduces the energy consumption of the sensor and inference model by up to 2×, compared to methods that only optimize the inference model. Jointly, they enable SolarML to run end-to-end gesture and audio inference on a battery-free tinyML platform by only harvesting solar energy for 30 and 57 seconds, respectively, in an office environment (500 lux). Source code is available at [1].**

I. Introduction

A key challenge for future Artificial Intelligence of Things (AIoT) is to balance the added computational intelligence and sustainability. To address this, the community has developed tinyML, a set of deep-learning models for low-power microcontrollers (MCUs) [2]–[11], pushing deep learning to the extreme edge devices, e.g., wearable devices [12], [13] and nano-drones [14]. Besides these advancements, more work is needed to further create a *solar-powered* battery-free tinyML platform. Solar-powered systems are now widely used in small-scale, resource-constrained applications, such as environmental monitoring, wearable devices, and smart agriculture [15]–[19]. However, the gap between the power produced by solar cells and the power consumed by tinyML models is still large. Small IoT devices can harvest power on the microjoule scale [15], [16], [20], while a single tinyML model inference may require energy ranging from millijoules [7], [21]–[23] to joules [24]. To further reduce this energy gap, we design a framework SolarML that optimizes jointly the *hardware* and *software* parameters of sensing and inference for solar-powered tinyML platforms.

Challenges. Until now, most studies focus on optimizing model inference energy, ignoring the energy consumed by hardware to detect and gather events for the model's input. When considering these hardware energy costs, we identify two challenges.

Challenge 1: *Detecting events can consume more energy than the inference model itself.* Fundamental to tinyML applications is to detect events in an energy-efficient way, like voice or gesture commands. Ideally, a solar-powered system should remain *off* when idle and *on* only when an event occurs. However, most systems continuously monitor sensors to improve responsiveness [12], [21], [25]. But it can consume significant amounts of energy, e.g., reaching up to 70% of total energy cost, as shown in Figure 1. Other systems go to deep sleep and use a low-power actuator to wake up the MCUs [5], [17], [22], [26]. This is more efficient, but it consumes non-trivial amounts of energy during long inactive periods: approximately 15% of total energy cost, as in [22], [26] (cf. Figure 1). Using mechanical switches is another option, but it is not user-friendly. They require physical press and have longer response time. The take-home message is that, beyond a point, further optimizing the inference model *alone* does not have an effect impact on the energy consumption of the overall system.

#	Ref.
1	[21]
2	[17]
3	[26]
4	[22]
5	Gesture
6	Audio

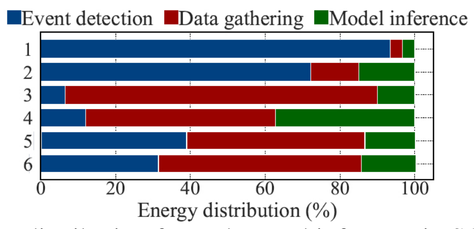


Fig. 1: Energy cost distribution for end-to-end inference in SOTAs. *Event-detection energy assumes a 3s wait time. Values for [21] to [22] are derived from their available data; gesture (#5) and audio (#6) are measured using μNAS [4] optimizations.*

Challenge 2: *The parameters used for data gathering have a fundamental impact on designing optimal deep learning models for solar-powered tinyML platforms.* Many studies use neural architectural search (NAS) to obtain small yet powerful deep learning models that can run on MCUs [3]–[8], [23], [27], [28]. Some of them optimize the network solely based on the memory and CPU constraints [3], [4], [8], [23], [27], while others also consider energy requirements [5]–[7], [28]. These studies are valuable, but the optimization focuses only on the network architecture or their energy cost without considering *sensing* energy costs. The sensing cost, however, could reach more than 50% of the total energy cost, as shown in Figure 1 for the gestures (#5) and audio recognition tasks (#6). Sensors cost different amounts of energy to provide different data formats, which impact the model's inference performance. Thus, to fully optimize solar-powered tinyML platform, we need to consider both the parameters of deep learning models and sensing.

Contributions. Denoting $E_{\mathcal{E}}$, $E_{\mathcal{S}}$, and $E_{\mathcal{M}}$, respectively, as the energy cost on event detection, signal processing, and model inference, prior work primarily focused on minimizing $E_{\mathcal{M}}$. We argue that the system should also optimize $E_{\mathcal{E}}$ (event detection), and all the parameters used to generate the input signal and run the model ($E_{\mathcal{S}} + E_{\mathcal{M}}$), instead of optimizing only $E_{\mathcal{M}}$.

Our approach requires a tight integration of different optimization phases, depending on the application and sensors used. Each type of sensor uniquely determines the event detection process and the optimizable parameters: audio (microphones), light (photodiodes), exposing widely different configurations. We evaluate our SolarML on a solar-powered tinyML platform through two applications: (i) gesture recognition with solar cells, and (ii) keywords spotting (KWS) with an on-board microphone (researched in tinyMLPerf [29]). Considering these applications, our work provides three main contributions:

Contribution 1: *An optimal design of solar event detectors.* Our platform uses solar cells not only for energy harvesting and sensing but also for event detection, reducing reliance on external components. Rather than using an external actuator to switch between deep sleep and active modes, the solar cells directly control this function. The system remains entirely off until a user's gesture activates it. This efficient design reduces power consumption by up to 10 times compared to conventional systems that use deep sleep mode and low-power actuators [15], [16], significantly lowering event detection costs ($E_{\mathcal{E}}$).

Contribution 2: *A method eNAS that jointly optimizes sensing and model parameters.* We design eNAS, a method optimizing both sensing and model parameters using genetic algorithm techniques. eNAS has three key innovations: First, it develops an energy model for hardware. Second, it corrects a commonly misused energy proxy equation, doubling energy estimation accuracy based on error rates. Third, rather than optimizing solely the network architecture ($E_{\mathcal{M}}$), it integrates sensing-related parameters, enabling optimization of both $E_{\mathcal{S}}$ and $E_{\mathcal{M}}$. eNAS outperforms SOTA methods across 20 different configurations, reducing energy cost by up to $2\times$ in gesture tasks and $1.5\times$ in audio tasks while preserving the accuracy.

Contribution 3: *Implementation and evaluation.* Our prototyped SolarML runs solely on data and energy gathered by solar cells. Evaluations with digit recognition and KWS applications show that it reduces energy consumption by an average of 48%. Thus, SolarML can run end-to-end inference of the digit recognition and KWS by harvesting energy for just 30 and 57 seconds, respectively, even under 500 lux lighting conditions.

II. Background & Related Work

An detailed example: energy consumption trace of end-to-end inference. Figure 2 illustrates the energy traces for the gesture recognition and KWS tasks, aligning with '#5' (gesture) and '#6' (audio) in Figure 1. Initially, the system remains in *deep sleep*, then *wakes up* and *samples* the audio or gesture. The sampling operates in a low-power tickless mode [12], which uses an external clock peripheral for sampling tasks instead of the whole CPU. After that, the system runs the *inference* and returns to deep sleep. In this case, $E_{\mathcal{E}}$ includes the energy costs related to deep sleep and wake-up, $E_{\mathcal{S}}$ captures the sampling cost, and $E_{\mathcal{M}}$ measures the inference cost. When the MCU sleeps only for one minute and we wake it up to perform

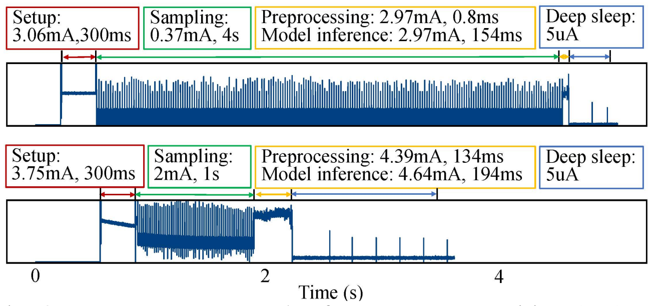


Fig. 2: Energy trace examples for gesture recognition (top) and KWS (bottom) using Qoitech OTII-ACE-PRO at 50kHz on an NRF52840 MCU with MbedOS.

gesture/audio recognition, $E_{\mathcal{M}}$ constitutes a mere 15% and 18% of the total energy cost, respectively. $E_{\mathcal{E}}$ accounts for 38% and 29%, while $E_{\mathcal{S}}$ takes 47% and 53%, respectively. Despite the lower power demands of $E_{\mathcal{E}}$ and $E_{\mathcal{S}}$ compared to $E_{\mathcal{M}}$, their longer execution times lead to significant energy use.

Event Detection. Event detection can be done in various ways. Some systems adopt constant monitoring, leading to high energy consumption. For example, PRO uses a headband to process ECG signals with a small inference model to detect epileptic seizures [12]. Others for face detection [17], [22] and related tasks [5], reduce power consumption by putting the system to deep sleep or using low-power sensors for activation. However, if the interval between events is long, deep sleep also costs a lot of energy. [30] explores piezo-trigger's passive characteristics for event detection, but this is limited to specific scenarios. In comparison, SolarML provides a novel mechanism to eliminate energy consumption during idle with solar cells.

Neural Architecture Search (NAS). It automates the search for optimal deep learning model architectures and parameters on a given dataset. It has three components: *(i) the search space*, encompassing various network parameters, *(ii) the search criteria*, bounding the search space based on memory, energy, or accuracy requirements, and *(iii) the search algorithm*, identifying the optimal hyperparameters within the valid search space.

Most NAS algorithms overlook *how sensing parameters affect energy consumption.* SpArSe uses a Bayesian optimization strategy to optimize model structures, focusing on model size and memory limits [27]. μNAS enhances the precision of the search space, rendering it suitable for *mid-tier* MCUs, emphasizing memory and latency [4]. Micronets handles latency and energy constraints with a differentiable approach [7]. HarvNet employs a multi-exit strategy to align accuracy with available energy [5]. While Micronets and HarvNet account for energy and latency, they overlook sensing parameters. TinyNAS adjusts sensing parameters by scaling input resolutions, but offers limited configurations and doesn't consider energy consumption.

NAS analyzes millions of configurations. *Directly measuring the energy use of each setup is time-consuming, so energy models are crucial for NAS.* Micronets and MCUnets build energy models using lookup tables recording latency and energy costs across layer configurations, but measuring all layer configurations is time-intensive. μNAS and HarvNet use Multiply-Accumulates (MACs) to estimate inference latency and energy cost, effective only when models share similar back-

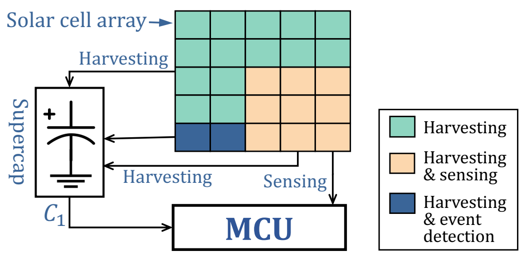


Fig. 3: System overview of our SolarML.

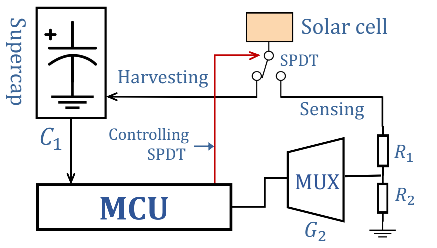


Fig. 4: Circuit for harvesting & sensing.

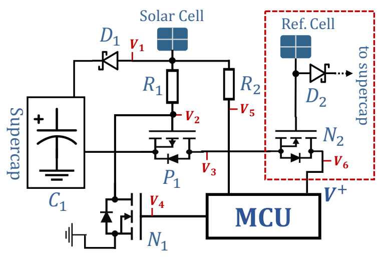


Fig. 5: Circuit for detecting events.

bones/layers [4], [5]. Our SolarML provides a more accurate energy model for inference and sensing.

III. SolarML Design

A. System Overview

Figure 3 shows the system overview of our SolarML. It uses solar cells to harvest energy and store it in a supercap; the MCU can run a tinyML model with this solar energy. Solar cells are also used for gesture sensing and event detection. Furthermore, we design an eNAS algorithm to optimize both sensing and inference parameters, significantly reducing MCU's energy cost. Beyond gesture sensing, our eNAS can support other tasks like audio sensing; we will demonstrate this later with an audio application using the MCU's onboard microphone.

Expanding roles of solar cells. The core hardware design of SolarML is the exploitation of solar cells for various roles. Instead of only using solar cells for either energy harvesting [13], [17], [22] or sensing [15], [16], we exploit the same solar cells for both tasks. Beyond this *dual use*, and motivated by the observation that traditional sensing systems may dedicate too much power on detecting events ($E_{\mathcal{E}}$), our SolarML introduces a *third use* of the solar cells: *detecting sensing events.* It works similar to a proximity sensor (PS) such as in [17], but in a passive manner. Thus, it is more energy efficient. In Figure 3, we thus see three types of solar cells (without loss of generality, we consider 25 solar cells). *All of them*, 25 cells, perform *energy harvesting*. The yellow ones, nine cells on the bottom right, can also perform *sensing*; and the blue ones, two cells on the bottom left, are also exploited for *event detection*.

e**NAS**. As shown in Section II, most NAS approaches focus on balancing accuracy and the energy used by model inference, but they often overlook the energy cost on sensing. In SolarML, we design *eNAS* to balance the accuracy and energy cost of both sensing and model inference. eNAS optimizes the parameters ranging from data sampling and pre-processing to model prediction. To optimize energy, we propose two energy models: *i) Sensing energy model,* considering sampling parameters such as channels, rate, resolution, quantization and parameters in the preprocessing phase. *ii) Inference energy model with layer-wise MACs,* precisely estimating a model's energy consumption.

Next, we detail the diverse roles of solar cells in SolarML. The proposed eNAS optimization will be detailed in Section IV.

B. Expanding Roles of Solar Cells

We show how solar cells perform energy harvesting and sensing dual functions, and expand their utility to event detection.

1) Energy Harvesting & Sensing: Prior studies attempt or achieve dual energy harvesting and sensing using solar cells as a single pixel [15], [16], [31], [32]. To achieve this goal, we employ a similar strategy as the study [33], using single-pole double-throw (SPDT) switch to separate the energy harvesting and sensing signals at each *sensing* solar cell, as shown in Figure 4. The solar cells operate in the energy-harvesting mode when no event is detected. In this mode, the SPDT switches connect all 9 *sensing* solar cells to the energy harvesting branch. For the two solar cells designated for the *event detection*, we connected them to the supercap through two blocking diodes (Schottky diodes) to prevent reverse current flow. The remaining 14 solar cells, solely for energy harvesting, are directly connected to the supercap. To maximize harvesting efficiency, our circuit connects all the solar cells in parallel to increase the charging power. When a sensing event is detected, the MCU is immediately powered on and checks if the supercap voltage is sufficient for inference ($V > V_\theta$). The MCU proceeds with the inference only when sufficient power is available. For KWS, analog switches continuously harvest power; for gesture tasks, the analog switches first open the sensing branch to generate the sensing signals. To implement it, we design a voltage divider circuit with two resistors, R_1 and R_2, and sample the sensing signal from the line connecting the two resistors [34]. These signals are then interconnected to a multiplexer to minimize the utilization of MCU pins.

2) Event Detecting: In sensing systems, detecting the event, i.e., when users start performing the gestures, often leads to significant energy waste. This is primarily due to the need for additional active sensors (such as proximity sensors [17]), or substantial overhead in software to detect the start of user gesture [12], [15], [21]. These methods are unsuitable for solar-powered systems due to their high energy demands. Our platform reuses the solar cells to detect events with a low overhead, *adding a third functionality to the cells.*

Like other proximity sensors, our event detector also senses when a user is nearby: A user initiates or terminates sensing by hovering over the designated solar cell. The basic circuit design is shown in Figure 5, relies on four crucial functions: *(i) Event detection.* When a user hovers over the solar cell to indicate the start of a gesture, voltage V_2 drops. This triggers the MCU to power up and sample the signals. For this purpose, we leverage a P-MOSFET (P_1) as a switch, controlling the connection between the supercap (C_1) and MCU. When V_2 drops, P_1 connects the MCU to the supercap. *(ii) Maintaining MCU Activity.* User hovers over the solar cell for a short time to indicate the start of the sensing input; that is, V_2 would regain a higher level, and thus P_1 would disconnect the MCU from the supercap. To keep P_1 on, we use a digital pin of the MCU to provide a control signal (V_4 will be pulled high) to an N-MOSFET (N_1) switch, which keeps V_2 connected to the ground, thus P1 can maintain the connection between the supercap and MCU. *(iii) End of sampling.* The system also needs to know when the gesture ends. For this, we add extra

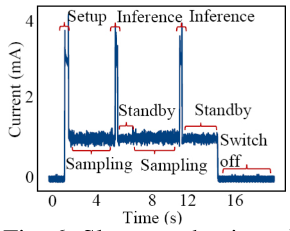


Fig. 6: Sleep mechanism.

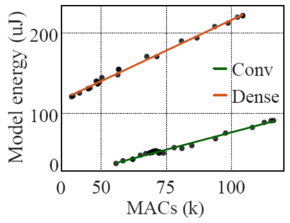


Fig. 7: Different layers' energy cost.

TABLE I: Comparison of energy estimation methods: Linear (LR), Logistic (LogR), Neural Regression (NR).

	Inference model				**Solar sampling model**		
Proxies	MACs [5]	Layer-wise MACs (*e*NAS)			n, r, b, q (Ref. Table II)		
Methods	LR	LR	LogR	NR	LR	LogR	NR
R^2	0.46	0.96	0.018	0.75	0.92	0.48	0.70

sense resistors (R_1, R_2) to the circuit and use the MCU to sample the voltage signal. This signal (V_5) indicates the ongoing sensing activity; once V_5 drops (solar cell being hovered over by the user again to indicate the gesture is finished), the MCU will stop the sampling of the voltage from the solar cells used for sensing, re-configure them to energy harvesting mode, and then continue to process the data and perform the model inference. *(iv) Handling weak-light conditions.* We add an additional N-MOSFET (N_2) switch and a reference solar cell. The N_2 remains non-conductive, preventing the supercap from powering up the MCU.

Figure 6 displays the efficacy of our event-detection design combined with sleep mechanism. First, the system exits the *off* mode when our event detection mechanism activates the MCU, which then remains continuously powered. The platform begins to sample data and stops when the event detector captures the ending signal. Subsequently, the system processes the first inference. Following this, the MCU enters a standby mode that preserves the system configuration in RAM while turning off the main CPU clock. The duration of this standby period depends on the energy spent during setup and the power used in standby. If the solar cell is hovered over again during this time, the system will resume and perform a second inference; otherwise, it powers down.

IV. *e*NAS: Energy-Efficient NAS

We propose *e*NAS, a hyperparameter search algorithm that balances energy consumption and accuracy. Our contributions include energy models and an optimization method.

A. Energy Models

1) Energy Model for Inference: Previous work shows that stacking different neural backbones or layers has different latency even with the same number of MACs [7]. Our measurements extend this to energy consumption: with 75k MACs, the *Dense* layer consumes 50 uJ, while the *Conv* layer consumes 175 uJ, a difference factor of 3.5, as shown in Figure 7. This suggests the need for layer-specific energy models. However, μNAS and HarvNet use a single energy model for the entire NAS operation, which can be written as $E_{\mathcal{M}} = a \cdot \text{MACs}_{\text{all}} + b$, where coefficients a and b are derived from linear regression with empirical data. To overcome that limitation, we propose a new method that uses *layer-wise MACs* rather than the total number of MACs. We empirically measure the energy of 300 models with different layers and numbers of MACs. Then, we fit the collected data into different regression methods on a per-layer basis. Table I presents the results. We observe that SOTA approach, which uses a single model, provides a low goodness-of-fit, 0.46. Among the three regression methods tested for our layered approach (linear, logistic, neural), linear regression is the best with a 0.96 goodness-of-fit. Thus, our energy model for inference is given by: $E_{\mathcal{M}} = a_1 \cdot \text{MAC}_{\text{AvgPool}} + a_2 \cdot \text{MAC}_{\text{MaxPool}} + a_3 \cdot \text{MAC}_{\text{D-Conv}} + a_4 \cdot \text{MAC}_{\text{Dense}} + a_5 \cdot \text{MAC}_{\text{Norm}} + a_6 \cdot \text{MAC}_{\text{Conv}} + b$, where parameters a_i and b are derived from 300 measurements.

2) Energy Model for Sensing: Some studies have tried to optimize the sensing parameters for a given model. However, they could only evaluate a few or a single parameter because they do not have energy models to perform an exhaustive, fine-grained search [6], [13], [15], [16], [21], [35]. For our gesture tasks, we consider the following sensing parameters: number of channels, rate, resolution, and quantization, as detailed in Table II. We conduct 300 measurements with random parameter values across these dimensions and analyze the data using linear, logistic, and neural regression methods. Table I reveals that linear regression achieves the highest accuracy in this task, with a fitness value of 0.92. For audio tasks, our energy model considers three parameters: window stripe, window duration, and the number of features. These parameters, commonly employed in audio preprocessing, are used to segment the speech signals into short-term frames. The corresponding search space for these parameters is presented in Table II. Following the same process in gesture tasks, we first collect 300 energy measurement data points and then establish energy equations through different regression equations. Our results show that the linear regression model also has the highest fit, with a coefficient of determination (R^2) of 0.99.

TABLE II: *e*NAS search space.

Tasks	**Sensing Parameters**	**Parameter Range**	**Morphisms**
Gesture recognition	Number of channels	$n \in [1,9]$	$n \pm 1$
	Sampling rate	$r \in$ [10, 200] Hz	$r \pm 2$
	Bit resolution	$b \in$ {int, float}	replace
	Quantization	$q_{\text{int}} \in [1,8]$, $q_{\text{float}} \in [9,32]$	$q_{\text{int}} \pm 1$, $q_{\text{float}} \pm 1$
KWS	Window stripe	$s \in [10,30]$	$s \pm 1$
	Window duration	$d \in [18,30]$	$d \pm 1$
	Number of features	$f \in [10,40]$	$f \pm 1$
Model hyperparameters space same as μNAS [4]			

B. Optimization Framework

Our optimization considers the hyperparameters of the sensing and inference parts and builds upon the area of genetic algorithms, which are widely used in the SOTA [4], [5], [36]. The problem is to maximize accuracy (A) while minimizing energy (E), with constraints on memory, MACs, and a minimum accuracy requirement. We consider the objective function: $\max\{A - \lambda(E - E_{\min})/(E_{\max} - E_{\min})\}$, where λ controls the trade-off between accuracy and energy, with values ranging from 0 (accuracy-focused) to 1 (energy-focused).

Our objective function is different from the ones used in μNAS and HarvNet. They use random scalarization to combine multiple objectives into a single goal [37]. The methods allow exploring the Pareto frontier, but there are no intuitive guidelines to set the λ_i, and the algorithm's answer is heavily influenced by these weights. Compared to μNAS, HarvNet combines accuracy and energy consumption into a single

Algorithm 1 *e*NAS Optimization.

```
1:  procedure OBTAIN OPTIMAL PARAMETERS($\lambda$)
2:      $population \leftarrow$ empty queue
3:      while $|population| < P$ do                          ▷ Phase 1
4:          $CAN \leftarrow$ RANDOMCANDIDATE()
5:          TRAINEVAL($CAN$) with constraints
6:          add $CAN$ to $population$
7:      $E\_max, E\_min \leftarrow$ FINDENERGY($population$)
8:      for $C$ cycles do                                     ▷ Phase 2
9:          $sample \leftarrow$ sample S candidates in population
10:         $parent \leftarrow$ best candidate in $sample$
11:         $child \leftarrow$ RANDOMMUTATE($parent$)
12:         if cycles mod $R$ is 0 then
13:             $parent \leftarrow$ best candidate in $sample$
14:             $child \leftarrow$ GRIDMUTATE($parent$)
15:         add $child$ to $population$ and remove $oldest$
        return the best candidate
```

objective function: $\max\{A/E\}$. But the lack of parameters does not allow exploring the Pareto frontier. Our objective function, however, requires estimated values for the maximum and minimum energy consumption of the overall system (E_{min} and E_{max}), which we do not know apriori. To solve this dependency, we propose a two-phase search algorithm.

1) Phase 1: broad exploration with random permutations: Initially, we set random parameters for the sensing and inference models. For each permutation i, we obtain its accuracy A_i and energy consumption E_i. At this stage, we apply the constraints regarding memory, computation, and accuracy, but we do *not* combine the accuracy A_i and energy E_i into our objective function. Our goal is to sample the search space broadly. After obtaining P permutations, we identify the minimum and maximum energy consumption: $E_{\text{min}} = \min\{E_i\}$ and $E_{\text{max}} = \max\{E_i\}$, where $i = \{1, \dots, P\}$. Since we do not perform an exhaustive search of the space, E_{min} and E_{max} may not be precise estimations but this does not affect our approach.

2) Phase 2: optimal exploration with mutations: After broad exploration, we take three steps to narrow our search towards the Pareto frontier. First, depending on the objective, we set the value of λ to focus on the permutation that maximizes accuracy, energy, or their trade-off. Second, we combine A_i and E_i of each permutation into our objective function. Third, we rank the results and choose the best permutation. With these steps, we identify the search space (Pareto frontier) that is likely to contain our optimal candidate. Next, we propose a customized search algorithm to navigate the sensing and inference dimensions. Prior studies show that minor adjustments to sensing parameters provide minimal performance gains if model parameters remain stable [6], [15]. Thus, we introduce a factor R to control sensing parameter adaptation, avoiding unnecessary computational costs. Given the limited range of sensing parameters, we perform a local grid search [38] to find the optimal candidates.

The final algorithm is shown in Algorithm 1. Phase 1, broad exploration, is in lines 3-6. Once P permutations are obtained, we estimate the remaining energy parameters in line 7 (E_{min} and E_{max}). Phase 2 begins with optimized mutations, evolving the model parameters each cycle (lines 9-11), while sampling parameters evolve every R cycles (lines 12-14).

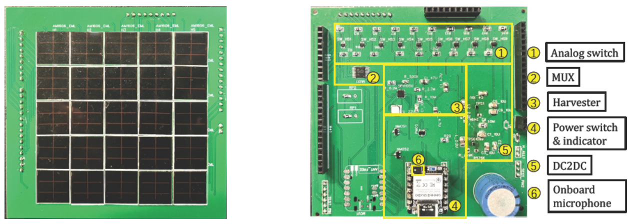


Fig. 8: Our SolarML Prototype: left) front side; right) backside.

V. Performance Evaluation

A. SolarML Prototype

Figure 8 shows the prototype of our SolarML, built on Xiao nRF52840 MCU. We use 13mm×13mm AM1606C solar cells for both harvesting and sensing, arranged in a 65mm×65mm array similar to other solar-powered systems [15], [22]. For energy harvesting, we chose the SPV1050 chip due to its compatibility with our solar cells, and connected it to a 1F supercap to meet the energy demands of tinyML applications [5]. Energy consumption is measured by Qoitech OTII-ACE-PRO analyzer. For sensing, we enable efficient switching between energy harvesting and sensing using ADG749 analog switches and the ISL84781 multiplexer. SI2309 and SI2304 MOSFETs are used for their high gate voltages, while the TPS61099 DC2DC module supplies 3.3V to the MCU.

B. Event Detection

We compare three event detection methods—proximity sensor (PS) [12], time-of-flight (ToF) sensor [17], and SolarGest [15]—with our approach, detailed in Table III. Proximity and ToF sensors emit light and measure reflections to detect events, covering ranges up to 100 mm and 4 m, respectively. In contrast, our method and SolarGest, which involve hovering over or covering the solar cell, detect events within a 20 mm range, sufficient for applications requiring close user interaction.

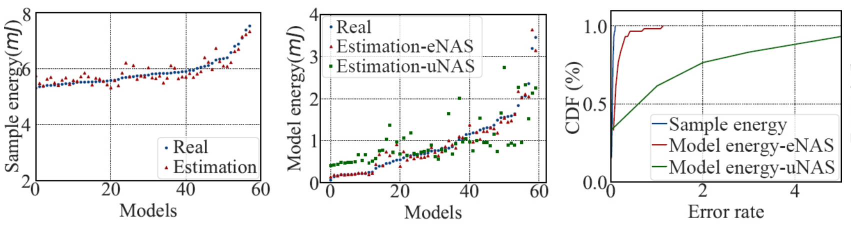
Our method outperforms others in response time; it does not require signal emission and processing, enabling a rapid 5 ms response. This is significantly faster than SolarGest, which necessitates a user to hover their hand for a second over the sensor. Our hardware-centric approach also ensures minimal power use—only 2μW in standby mode and up to 28μW during active phases, far more efficient than other sensors that consume between 7 and 1000μW. Despite a passive circuit design, our method supports complex gesture recognition, markedly outperforming others in energy efficiency and capability. Considering a scenario where the event detector waits for 5 seconds and then activates, our energy consumption is 10× lower than SolarGest, 7× and 4× less than ToF and PS sensors, respectively.

C. Energy Models for Sensing and Inference

Now, we present the evaluation of our energy models. The evaluation results of our sensing energy model are given in Figure 9(a). We run the measurement 60 times, and measure

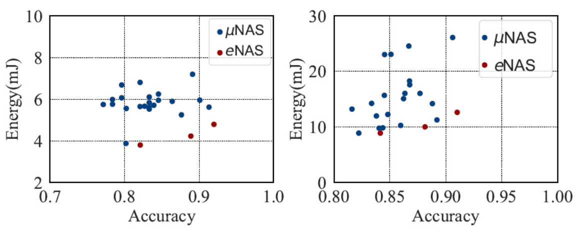
TABLE III: Event detection comparison (PS: Proximity sensor).

Metrics	PS [12]	ToF [17]	SolarGest [15]	SolarML
Sensing range (mm)	0-100	0-4000	0-20	0-20
Response time (ms)	10-700	20-1000	>1000	5
Standby power (μW)	7	10-30	not avail.	≈2
Working power (μW)	1000	1000	20	7.5-28
5-s work energy (μJ)	45-735	70-1150	100	≈10



(a) Sensing energy model (b) Inference energy model (c) CDF of error rate

Fig. 9: Evaluations of the energy models for sensing and inference.



(a) Digits recognition (b) Keyword Spotting

Fig. 10: Performance of our eNAS.

the real energy consumption on sampling (blue dots in the figure). We also plot the estimated energy consumption using our energy model. The real energy consumption matches well with the estimated values, with an average error of 3.1%. We also plot the CDF of the errors in Figure 9(c). More than 90% of the models fall below an error rate of 6%. The evaluation results of our inference energy model for the inference are given in Figure 9(b). We run the measurement also for 60 times. First, we can observe that the energy consumption of different models varies greatly from one to another, ranging from $10^{-2}mJ$ to $10^{-1}mJ$, and even can go beyond $3mJ$. Our model's estimates (red triangles) closely match the actual values, with an average error of 12.8%, a significant improvement over μNAS's 76.9% average error. The CDF shows that more than 90% of our model estimates fall below a 30% error rate, compared to μNAS's error rate of up to 400% to achieve similar coverage. This precise energy estimation for inference and sensing is key to our optimization strategy, enhancing overall system performance.

D. Application Evaluation

We evaluate the performance of our proposed eNAS framework, compared with the SOTA μNAS [4], through two tasks: digit recognition and KWS. For a fair comparison as outlined in [39], we leverage the open-source code of μNAS in our implementation. Besides, both eNAS and μNAS were executed under the same settings and datasets with the same constraints and search conditions: population size of 50, sample size of 20, and 150 evolutionary rounds. Both frameworks were also subjected to the same memory and MAC constraints—100KB and 30M max MACs respectively—and error rates, with KWS at 0.3, and digit gestures at 0.25. These configurations are commonly applied in NAS algorithms [3], [4].

In μNAS, the search space does not involve sensing hyperparameters. To thoroughly compare μNAS with eNAS, we test the performance of μNAS under different combinations of sensing hyperparameters. For this, we randomly combine 20 sets of sensing hyperparameters and employ them in the μNAS algorithm. For eNAS, the tradeoff between the accuracy and energy consumption of neural architectures is controlled by the parameter λ. In our evaluation, we consider three λ values: 0 (prioritizing high accuracy), 1 (prioritizing low energy consumption), and 0.5 to explore the Pareto frontier. Additionally, we empirically set the variable t to 20, based on our hardware capabilities and guided by practical experience and principles similar to learning rate scheduler [40].

Application 1: Digits recognition. The results are shown in Figure 10(a). First, we observe that using different λ, eNAS can search for the best neural architectures that trade between model accuracy and energy consumption. Given the same accuracy, eNAS spends much less energy when compared to that of μNAS. For example, for a targeted accuracy of 0.82, μNAS spends more than $1.5\times$ energy on average, compared to eNAS.

Application 2: Audio-based Keyword Spotting (KWS). The results are shown in Figure 10(b). Similarly, eNAS leads to the most energy-efficient models under various energy-accuracy trade-offs. Compared to the SOTA μNAS, eNAS improves the energy cost and accuracy. For example, given an energy budget of 10 mJ, eNAS leads to a model with 88% accuracy, higher than the 86% achieved by the model searched from μNAS. If the aim is to maximize accuracy ($\geq$90%), the best model from μNAS consumes $2.1\times$ more energy compared to our eNAS.

Overall, these applications show a consistent and robust performance of eNAS, demonstrating also the generalizability of our solutions. Regarding the complexity, they take 0.5 to 2 GPU-days. Here, 1 GPU-day means the search is conducted for a whole day using an A10 GPU.

Energy costs and harvesting time of end-to-end inference. We then evaluate the end-to-end energy costs of our SolarML (eNAS is part of it). SolarML combines event detection with NAS design, resulting in significant energy savings. We calculate the average energy consumption for the NAS design across varying settings (eNAS: $\lambda = 0, 0.5, 1$; μNAS: the three accuracy points closest to eNAS). For digit recognition, SolarML uses about 6660 μJ, while KWS consumes 12746 μJ. Compared to the SOTA PS [12] system with μNAS, which uses 8468 μJ and 18842 μJ respectively, our solution achieves impressive energy savings of 27% for the digit recognition and 48% for KWS.

For the harvesting time, our system needs just 31 seconds for digit recognition and 57 seconds for KWS in a typical office environment with 500 lux lighting. Near a window, with light intensity of 1000 lux or more, the harvesting time drops to 19 and 36 seconds, respectively. Even in dimmer conditions such as 250 lux, our eNAS models require only one to two minutes of energy harvesting. This high performance is possible due to the end-to-end optimization of $E_{\mathcal{E}} + E_{\mathcal{S}} + E_{\mathcal{M}}$ in our SolarML.

VI. Conclusion

In this paper, we designed SolarML, a solar-powered tinyML platform using solar cells for energy harvesting, gesture sensing and event detection. By repurposing solar cells for hardware-based event detection, we significantly reduced the energy consumption. Additionally, we designed eNAS to optimize both sensing and inference parameters to balance the energy cost and accuracy. Evaluations on digit recognition and audio-based KWS applications show the effectiveness of our solution.

Acknowledgment: This work is partly supported by the NWO *LuxSenz* project, funded through the TOP Grant (Module 1) under project number 612.001.854.

References

[1] "Solarml: Optimizing sensing and inference for solar-powered tinyml platforms." [Online]. Available: https://github.com/haoliu001/SolarML

[2] P. Warden and D. Situnayake, *TinyML: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

[3] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on iot devices," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[4] E. Liberis, L. Dudziak, and N. D. Lane, "μNAS: Constrained neural architecture search for microcontrollers," in *Proceedings of the Workshop on Machine Learning and Systems (EuroMLSys)*, 2021.

[5] 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 ACM Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2023.

[6] A. Montanari, M. Sharma, D. Jenkus, M. Alloulah, L. Qendro, and F. Kawsar, "Eperceptive: Energy reactive embedded intelligence for batteryless sensors," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2020.

[7] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, and P. Whatmough, "MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers," in *Proceedings of the Conference on Machine Learning and Systems (MLSys)*, 2021.

[8] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "Memory-efficient patch-based inference for tiny deep learning," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[9] H. Ren, D. Anicic, and T. A. Runkler, "Tinyol: Tinyml with online-learning on microcontrollers," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2021.

[10] F. Daghero, A. Burrello, C. Xie, M. Castellano, L. Gandolfi, A. Calimera, E. Macii, M. Poncino, and D. J. Pagliari, "Human activity recognition on microcontrollers with quantized and adaptive deep neural networks," *ACM Transactions on Embedded Computing Systems*, 2022.

[11] H. Liu, H. Ye, J. Yang, and Q. Wang, "Through-screen visible light sensing empowered by embedded deep learning," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021.

[12] N. Pham, H. Jia, M. Tran, T. Dinh, N. Bui, Y. Kwon, D. Ma, P. Nguyen, C. Mascolo, and T. Vu, "Pros: An efficient pattern-driven compressive sensing framework for low-power biopotential-based wearables with on-chip intelligence," in *Proceedings of the ACM Conference on Mobile Computing and Networking (MobiCom)*, 2022.

[13] H. Truong, S. Zhang, U. Muncuk, P. Nguyen, N. Bui, A. Nguyen, Q. Lv, K. Chowdhury, T. Dinh, and T. Vu, "Capband: Battery-free successive capacitance sensing wristband for hand gesture recognition," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2018.

[14] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "Ultra low power deep-learning-powered autonomous nano drones," in *Proceedings of IEEE/RSJ IROS*, 2018.

[15] D. Ma, G. Lan, M. Hassan, W. Hu, M. B. Upama, A. Uddin, and M. Youssef, "Solargest: Ubiquitous and battery-free gesture recognition using solar cells," in *Proceedings of the ACM Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[16] M. M. Sandhu, S. Khalifa, K. Geissdoerfer, R. Jurdak, and M. Portmann, "Solar: Energy positive human activity recognition using solar cells," in *Proceedings of the IEEE Conference on Pervasive Computing and Communications (PerCom)*, 2021.

[17] M. Giordano, P. Mayer, and M. Magno, "A battery-free long-range wireless smart camera for face detection," in *Proceedings of ACM ENSsys Workshop*, 2020.

[18] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, and K. Huang, "A survey on smart agriculture: Development modes, technologies, and security and privacy challenges," *IEEE/CAA Journal of Automatica Sinica*, 2021.

[19] V. Tsoukas, E. Boumpa, G. Giannakas, and A. Kakarountas, "A review of machine learning and tinyml in healthcare," in *Proceedings of the 25th Pan-Hellenic Conference on Informatics*, 2021.

[20] Y. Li, T. Li, R. A. Patel, X.-D. Yang, and X. Zhou, "Self-powered gesture recognition with ambient light," in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2018.

[21] A. Kiaghadi, J. Huang, S. Z. Homayounfar, T. Andrew, and D. Ganesan, "Fabtoys: Plush toys with large arrays of fabric-based pressure sensors to enable fine-grained interaction detection," in *Proceedings of Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2022.

[22] P. Jokic, S. Emery, and L. Benini, "Battery-less face recognition at the extreme edge," in *Proceedings of IEEE International New Circuits and Systems Conference*, 2021.

[23] S. S. Saha, S. S. Sandha, S. Pei, V. Jain, Z. Wang, Y. Li, A. Sarker, and M. Srivastava, "Auritus: An open-source optimization toolkit for training and development of human movement models and filters using earables," in *Proceedings of ACM IMWUT/UbiComp*, 2022.

[24] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-yolo: spiking neural network for energy-efficient object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[25] J. Moosmann, P. Bonazzi, Y. Li, S. Bian, P. Mayer, L. Benini, and M. Magno, "Ultra-efficient on-device object detection on ai-integrated smart glasses with tinyissimoyolo," *arXiv:2311.01057*, 2023.

[26] A. Sabovic, M. Aernouts, D. Subotic, J. Fontaine, E. De Poorter, and J. Famaey, "Towards energy-aware tinyml on battery-less iot devices," *Internet of Things*, 2023.

[27] I. Fedorov, R. Adams, M. Mattina, and P. Whatmough, "Sparse: Sparse architecture search for cnns on resource-cons-trained microcontrollers," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[28] R. Groh and A. M. Kist, "End-to-end evolutionary neural architecture search for microcontroller units," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023.

[29] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau *et al.*, "MLPerf Tiny Benchmark," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[30] M. Afanasov, N. A. Bhatti, D. Campagna, G. Caslini, F. M. Centonze, K. Dolui, A. Maioli, E. Barone, M. H. Alizai, J. H. Siddiqui, and L. Mottola, "Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2020.

[31] S. Das, A. Sparks, E. Poves, S. Videv, J. Fakidis, and H. Haas, "Effect of sunlight on photovoltaics as optical wireless communication receivers," *Journal of Lightwave Technology*, 2021.

[32] M. S. Mir, B. G. Guzman, A. Varshney, and D. Giustiniano, "Passivelifi: Rethinking lifi for low-power and long range rf backscatter," in *Proceedings of the ACM Conference on Mobile Computing and Networking (MobiCom)*, 2021.

[33] R. H. Venkatnarayan and M. Shahzad, "Gesture recognition using ambient light," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2018.

[34] A. Varshney, A. Soleiman, L. Mottola, and T. Voigt, "Battery-free visible light sensing," ser. VLCS '17, 2017.

[35] M. Cui, B. Xie, Q. Wang, and J. Xiong, "Dancingant: Body-empowered wireless sensing utilizing pervasive radiations from powerline," in *Proceedings of the ACM Conference on Mobile Computing and Networking (MobiCom)*, 2023.

[36] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, 2019.

[37] B. Paria, K. Kandasamy, and B. Póczos, "A flexible framework for multi-objective bayesian optimization using random scalarizations," in *Uncertainty in Artificial Intelligence*. PMLR, 2020.

[38] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[39] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *arXiv preprint arXiv:2101.09336*, 2021.

[40] R. Ge, S. M. Kakade, R. Kidambi, and P. Netrapalli, "The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares," *Advances in neural information processing systems*, 2019.