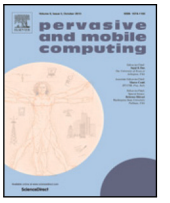
Pervasive and Mobile Computing 119 (2026) 102208

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



A cascade framework for on-device uncertainty-aware event detection on microcontrollers

Hong Jia [a],*, Young D. Kwon [a], Dong Ma [a], Nhat Pham [b], Lorena Qendro [c], Tam Vu [d], Cecilia Mascolo [a]

[a] University of Cambridge, Cambridge, UK
[b] Cardiff University, UK
[c] Nokia Bell Labs, UK
[d] Dartmouth College, USA

ARTICLE INFO

Keywords:
Uncertainty
Event detection
Efficiency
Microcontrollers

ABSTRACT

Pervasive sensing enables diverse wearable event detection (WED) applications, but deploying machine learning models on resource-constrained microcontrollers (MCUs) poses significant challenges, particularly in ensuring prediction reliability under data shifts or out-of-distribution (OOD) inputs. While Uncertainty quantification methods offer a way to assess this reliability, many are computationally prohibitive for MCUs, and detecting multiple events concurrently further exacerbates resource constraints. Addressing these combined challenges, this paper presents an uncertainty and resource-aware framework designed for reliable and efficient multi-event WED on MCUs, significantly extending our preliminary work.

The proposed framework achieves this by integrating Evidential Deep Learning (EDL) for efficient, single-pass uncertainty estimation with a novel cascade learning architecture. This architecture promotes resource efficiency via: (i) *intra-event* sharing using uncertainty-aware early exits within a staged model (shallow, medium, deep), allowing simpler samples to terminate inference earlier; and (ii) *inter-event* sharing using a multi-head design where multiple event detectors share a common backbone, minimizing overhead. System efficiency is further enhanced through MCU-specific optimizations, including targeted architecture search, quantization, efficient uncertainty operator implementation using standard TensorFlow Lite Micro (TFLM) operations, and library footprint reduction.

We conducted extensive experiments on four distinct wearable datasets (Oesense, KWS, ECG5000, and HHAR) and two MCU platforms (STM32F446ZE, STM32H747XI), comparing the proposed framework against strong baselines including Deep Ensembles and Vanilla EDL. Results demonstrate the proposed framework's effectiveness, achieving competitive accuracy and uncertainty performance (e.g., up to 22% lower NLL than data augmentation) while drastically reducing resource consumption, offering up to 8.64× faster inference, up to 8.57× lower energy use, and 55% smaller memory footprint compared to ensemble methods. The proposed framework enables the deployment of reliable, uncertainty-aware multi-event detection on a wider range of low-power MCUs.

* Corresponding author.
E-mail address: h.jia.cam@gmail.com (H. Jia).

https://doi.org/10.1016/j.pmcj.2026.102208
Received 5 January 2025; Received in revised form 30 April 2025; Accepted 10 March 2026
Available online 13 March 2026

1. Introduction

The proliferation of pervasive, low-power sensing technologies enables continuous collection and analysis of diverse human physiological signals. Machine learning (ML), particularly deep learning (DL), leverages these signals for a wide array of wearable event detection (WED) applications, ranging from stress monitoring [1] and blood pressure estimation [2] to identifying respiratory conditions [3]. Executing such ML models directly on resource-constrained microcontrollers (MCUs) has garnered significant interest, offering enhanced user privacy and reduced latency compared to cloud-offloading, especially where network connectivity is unreliable [4]. However, the stringent limitations in memory, processing power, and energy budget inherent to MCUs (Fig. 1) present substantial obstacles to designing and deploying efficient and effective WED systems [4].

A critical challenge, often overlooked in the pursuit of pure classification accuracy, is ensuring the *reliability* of predictions [5], a paramount concern in domains like mobile health. Prediction reliability can be formally assessed through *uncertainty quantification*, which measures the model's confidence in its outputs [6]. Real-world factors like hardware variability, environmental fluctuations, inconsistent data collection practices, or sensor degradation inevitably cause distribution shifts between training data and operational data (often termed *data uncertainty*) or lead to encounters with previously unseen data types (*model uncertainty* [7]). Such shifts can significantly impair the trustworthiness of deployed WED models.

This paper presents the proposed framework, an uncertainty and resource-aware framework for microcontrollers, significantly extending our preliminary work introduced in [8]. While various methods exist for uncertainty quantification, many, like Bayesian Neural Networks (BNNs) [9], impose prohibitive computational costs [10], largely due to the overhead of representing weights as full distributions and the computationally intensive sampling or approximation techniques often required for inference. Even approximations such as Monte Carlo Dropout (MCDP) [11] or Deep Ensembles [7], although improving accuracy, still demand multiple inference passes and substantial memory, rendering them largely unsuitable for typical MCUs. Conversely, simpler deterministic approaches often compromise accuracy [12]. Consequently, integrating reliable uncertainty estimation without overwhelming MCU resources remains a key hurdle for trustworthy on-device WED.

Furthermore, deploying models for *multiple* distinct events concurrently on a single MCU introduces additional efficiency challenges. A common practice involves using separate, optimized models for each event type [13] to maintain modularity. However, this multiplies the resource demands. For instance, analyzing electrocardiogram (ECG) signals for different heart conditions (e.g., detecting normal vs. premature ventricular contractions) would traditionally require two independent models. Performing multiple inferences, each potentially including uncertainty estimation, can quickly exhaust the constrained memory and energy budget of an MCU.

To overcome these combined challenges, the proposed framework integrates Evidential Deep Learning (EDL) with a novel cascade learning architecture optimized for MCUs. EDL allows the model to directly output a probability distribution (parameterized by evidence) rather than just a point prediction, enabling efficient uncertainty quantification within a single forward pass (Section 4). Our cascade architecture facilitates resource sharing both within a single event's model (intra-event) and across multiple event models (inter-event). For intra-event efficiency, models are structured into shallow, medium, and deep stages, sharing initial layers. An uncertainty-based early-exit mechanism allows simpler samples to terminate inference at earlier stages, saving computation (Section 5.1). For inter-event efficiency, multiple event detectors share the same core feature extraction backbone, requiring only separate classification "heads", minimizing memory overhead for adding new event types (Section 5.2).

This journal article extends the preliminary work presented in [8] by providing: (i) a more thorough exposition of the Evidential Deep Learning foundations relevant to resource-constrained uncertainty estimation (see Section 4.1), (ii) expanded discussion on the practical implementation challenges and optimizations for the uncertainty operator on MCUs (Section 6), (iii) evaluation using four datasets including an additional dataset (ECG5000) representing time-series physiological signals and comparison against an additional baseline (Vanilla EDL) to provide a broader comparative context (Section 7 & 8), (iv) a more detailed analysis of the trade-offs associated with selecting uncertainty thresholds (Section 8.2), and (v) an expanded analysis of feature extraction overheads on MCU platforms ().

System efficiency is further enhanced through pragmatic implementation techniques (Section 6). We employ an efficient architecture search strategy to identify suitable model structures tailored for MCUs [14], utilize quantization to reduce model size, and perform targeted optimizations on the TFLM library to minimize its memory footprint. Comprehensive experiments on four distinct datasets and two MCU platforms validate the effectiveness of the complete proposed framework.

In summary, the primary contributions of this work are:

- A novel cascade learning architecture employing uncertainty-driven early exits (intra-event) and multi-head feature sharing (inter-event) for efficient multi-event detection on MCUs (Section 5).
- The adaptation and integration of Evidential Deep Learning for efficient and reliable uncertainty quantification directly on MCUs, enabling single-pass prediction and uncertainty estimation (Section 4).
- Practical system optimizations including architecture search, quantization, and library tuning tailored for deploying uncertainty-aware models on resource-constrained hardware (Section 6).
- Extensive empirical validation on four distinct wearable datasets (representing audio, ECG, and motion sensing) and two MCU platforms, demonstrating significant improvements over baseline uncertainty methods in terms of inference speed (up to 8.64×), energy efficiency (up to 8.57×), and memory footprint (up to 55% reduction), while achieving competitive accuracy and uncertainty performance (Section 7–8).

	Mobile ML	TinyML		
Platform		STM32F205VB	STM32F446ZE	STM32H743VIT6
SRAM	6GB	64KB	128KB	1MB
eFlash	128GB	128KB	512KB	2MB
Power	20W	0.2W	0.8W	1.2W
Price	~$1000	$2	$3	$9

Fig. 1. Memory and power comparison between a typical mobile phone and MCUs.

2. Related works

This section contextualizes our work within the literature concerning machine learning on MCUs, event detection strategies for resource-constrained devices, and approaches for efficient uncertainty estimation.

Tiny machine learning on MCUs. The field of Tiny Machine Learning (TinyML) [14] specifically targets the execution of ML models on deeply embedded systems like MCUs. A significant body of research focuses on optimizing neural network architectures and training procedures to meet the stringent constraints of limited memory, low energy consumption, reduced computational operations (FLOPs) [4], and slower processor speeds [15]. However, the predominant focus remains on maximizing classification accuracy, often neglecting the crucial aspect of prediction reliability via uncertainty quantification. Our work diverges by explicitly integrating uncertainty estimation into the TinyML workflow for WED, aiming for trustworthy alongside efficient predictions.

Event detection on resource-constrained devices. Event detection using wearable sensor data is an active research area, spanning modalities such as imaging [16], audio [17], and electrocardiography (ECG) [18]. Many existing WED systems, however, treat the wearable primarily as a data acquisition device, offloading computationally intensive tasks like signal pre-processing, feature extraction, and ML inference to more powerful platforms, including cloud servers [3,19], desktop GPUs [20], smartphones [1], or less constrained IoT devices [21]. This reliance on external processing can introduce undesirable latency, compromise user privacy, and fail in scenarios with intermittent connectivity. In contrast, our research concentrates on enabling comprehensive, end-to-end WED computation directly on the MCU, necessitating highly efficient and lightweight ML models capable of operating reliably within severe resource limitations.

Efficient uncertainty estimation. While uncertainty estimation is vital for reliability, conventional methods pose challenges for MCUs. Bayesian Neural Networks (BNNs), though theoretically robust, typically involve computationally expensive operations like Markov Chain Monte Carlo (MCMC) sampling or variational inference, exceeding MCU capabilities [9]. Approximation techniques like MC Dropout [11] and Deep Ensembles [7] reduce the burden but still necessitate multiple forward passes during inference, leading to significant increases in latency and memory usage. Furthermore, implementing dropout layers efficiently or storing multiple ensemble members strains MCU resources. Other promising directions include methods based on normalizing flows [22], spectral normalization [23], or specialized stochastic layers [10], but these often introduce operations or require custom libraries not readily supported by standard TinyML frameworks like TFLM. Some approaches, like deterministic uncertainty methods [12] or certain non-Bayesian techniques [24], offer single-pass efficiency but at the cost of reduced uncertainty quality or accuracy compared to ensembles. Knowledge distillation [25] can transfer ensemble knowledge to a single model but typically requires access to representative out-of-distribution (OOD) data, which is often unavailable in practical WED scenarios. The proposed framework employs EDL a non-Bayesian approach that provides uncertainty estimates in a single forward pass, specifically adapted and optimized here for the multi-event, resource-constrained context of WED on MCUs, representing a novel contribution to reliable TinyML.

3. Proposed framework overview

The proposed framework encompasses two primary stages: offline model training (Section 4- Section 5) and on-device deployment and inference (Section 6), depicted conceptually in Fig. 2. During inference on the MCU, input features pass through the cascade stages (shallow, medium, deep), with uncertainty evaluated at intermediate exits to potentially terminate computation early based on a predefined threshold. The training stage focuses on constructing models that are both accurate and uncertainty-aware, utilizing: (1) Evidential Deep Learning (EDL) for efficient single-pass uncertainty quantification, and (2) A cascade learning structure enabling resource sharing through uncertainty-based early exits for single events (intra-event sharing) and multi-head feature sharing for concurrent events (inter-event sharing). The deployment stage then addresses the practicalities of running these models on MCUs, including: (1) leveraging multi-tenancy concepts [26] for dynamic memory sharing among model components (e.g., cascade stages), and (2) applying model and library optimizations for maximal efficiency.

Operationally, the process begins with wearable sensors capturing raw signal streams corresponding to potential events. Appropriate features (e.g., Mel-frequency cepstral coefficients (MFCC) for audio) are extracted on-device. These features are then fed into the core proposed framework's model. This model employs evidential reasoning via EDL, trained using a one-vs-all strategy (Section 4), to produce both the event prediction and an associated uncertainty score, indicating prediction reliability. The model architecture itself is structured as a cascade (Section 5). For detecting a single type of event, the network is divided into sequential shallow, medium, and deep stages. This facilitates intra-event sharing, where early exits allow inference to terminate prematurely for

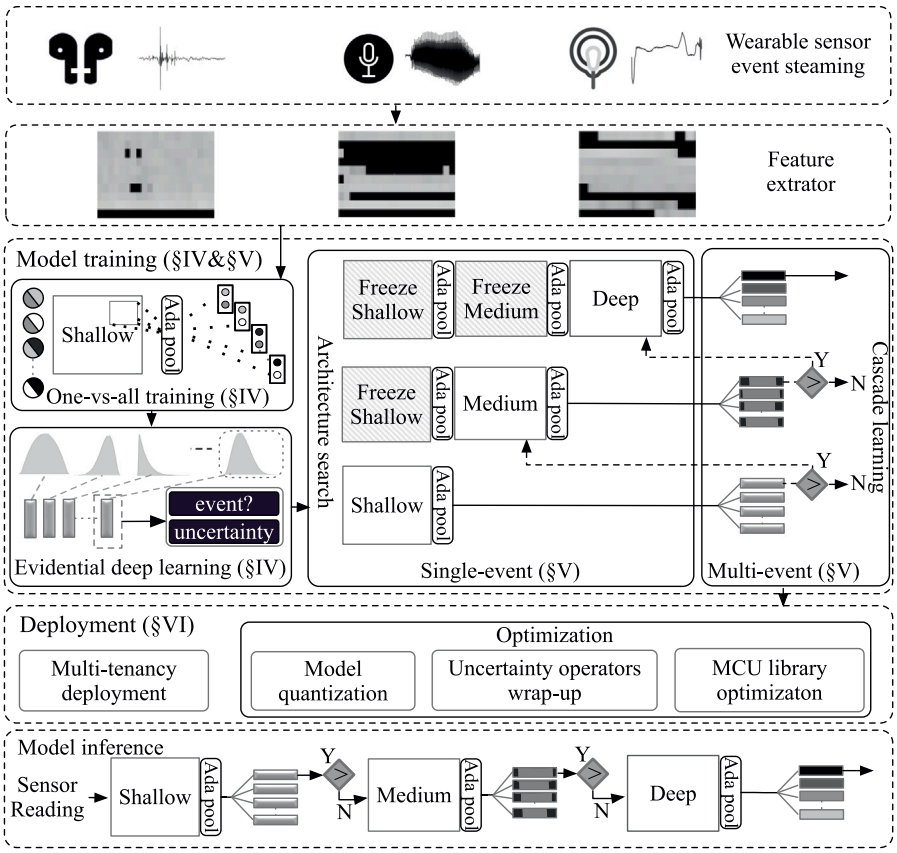


Fig. 2. Overview of the proposed end-to-end framework. Sensor data undergoes feature extraction, feeding into the model training stage (Section 4 & Section 5). This stage uses EDL for uncertainty-aware training via a one-vs-all strategy, architecture search to define efficient models, and cascade learning to create a Shallow-Medium-Deep structure enabling single and multi-event detection with early exits. Trained models are optimized for deployment (Section 6) via quantization and library tuning. Finally, the inference stage utilizes the deployed cascade, using uncertainty estimates to efficiently process sensor readings on MCUs.

'easy' samples based on their uncertainty, conserving resources. For detecting multiple event types from the same input signal, the framework employs inter-event sharing, where all event detectors reuse the same core cascade backbone for feature extraction, differing only in their final classification heads. Complementing the core modeling approach, we implement several efficiency enhancements (Section 6), including automated model architecture search performed offline, post-training model quantization, careful implementation of the uncertainty calculation using standard MCU library operators, and targeted optimization of the underlying TFLM library.

4. Efficient uncertainty quantification

This section details our adaptation of Evidential Deep Learning (EDL) for efficient and reliable uncertainty quantification within the demanding constraints of WED on MCUs. The goal is to develop a model capable of single-pass inference that yields both an event prediction and a meaningful measure of its associated uncertainty.

4.1. Evidential Deep Learning foundations

EDL [24] provides a framework for training deterministic neural networks to output parameters of a higher-order probability distribution over the classification outcome, rather than just point predictions or simple softmax probabilities. For a C-class classification problem, given an input x^i, an EDL model outputs the parameters $\boldsymbol{\alpha}^i = [\alpha_1^i, \alpha_2^i, \ldots, \alpha_C^i]$ of a Dirichlet distribution $Dir(\boldsymbol{\alpha}^i)$. These parameters, often called concentration parameters or evidence, quantify the amount of support gathered from the input for each class. Higher evidence values for a class correspond to higher confidence. The Dirichlet distribution serves as a conjugate prior to the Categorical distribution representing the class probabilities.

From the evidence vector $\boldsymbol{\alpha}^i$, EDL allows deriving the belief mass $\boldsymbol{b}^i = [b_1^i, b_2^i, \ldots, b_C^i]$ assigned to each class and the overall uncertainty. The total evidence is the Dirichlet strength $S^i = \sum_{c=1}^{C} \alpha_c^i$. The belief mass for class c is calculated as:

$$b_c^i = (\alpha_c^i - 1)/S^i, \tag{1}$$

where $\alpha_c^i \geq 1$. Intuitively, belief masses represent the assigned probability mass excluding the mass assigned to uncertainty. The overall uncertainty u^i (specifically, vacuity or ignorance) is then derived from the total evidence:

$$u^i = C/S^i, \tag{2}$$

A higher total evidence S^i leads to lower uncertainty u^i. The class prediction $\hat{y}^i$ is typically taken as the class with the highest expected probability under the Dirichlet distribution:

$$\hat{y}^i = \arg\max_c E[p_c^i|\boldsymbol{\alpha}^i] = \arg\max_c(\alpha_c^i/S^i), \tag{3}$$

Training an EDL model typically involves minimizing a specialized loss function. A common approach, adapted from [24], uses the expected cross-entropy under the Dirichlet distribution, potentially augmented with a regularizer based on the distribution's entropy or variance to prevent the model from becoming overconfident on incorrect predictions. For instance, a Type II Maximum Likelihood (Empirical Bayes) loss minimizes the negative logarithm of the model evidence:

$$\mathcal{L}(\boldsymbol{\alpha}^i, y^i) = \sum_{c=1}^{C} y_c^i(log(S^i) - log(\alpha_c^i)), \tag{4}$$

where y^i is the one-hot encoded true label. Other loss variants exist, often incorporating a term to explicitly penalize predictions with high uncertainty on correctly classified samples or low uncertainty on misclassified samples. The choice of activation function for the final layer outputting $\boldsymbol{\alpha}^i$ is crucial; it must ensure $\alpha_c^i > 0$ (often $\alpha_c^i \geq 1$). Functions like ReLU, Softplus, or Exponential are commonly used, though availability within specific MCU libraries (like TFLM) can constrain this choice, as discussed in Section 6.

4.2. Efficient evidential modeling for WED on MCUs

Directly applying the general EDL formulation for multi-class classification to WED tasks, especially in a multi-event scenario, poses efficiency challenges for MCUs. Each event detection is typically framed as a binary classification problem (event vs. non-event). Naively deploying separate EDL models for each event would multiply resource usage.

To address this, we tailor the EDL framework for efficient binary classification suitable for WED and integrate it with resource-sharing mechanisms. For a binary task (event c vs. non-event), the Dirichlet distribution simplifies to a Beta distribution, characterized by two positive parameters, α_c^i and β_c^i. These represent the evidence gathered for the positive (event) class and the negative (non-event) class, respectively.

Efficient EDL Modeling via Beta Distribution. For a single event c, we model the event probability p_c^i using a Beta distribution parameterized by $\alpha_c^i, \beta_c^i > 0$:

$$\begin{aligned} \mathrm{P}(p_c^i \mid x^i; \theta_c) &= \mathrm{Beta}(p_c^i \mid \alpha_c^i, \beta_c^i) \\ &= \frac{1}{B(\alpha_c^i, \beta_c^i)} p^{\alpha_c^i - 1}(1-p)^{\beta_c^i - 1}, \end{aligned} \tag{5}$$

where $B(\cdot,\cdot)$ is the Beta function. The neural network is trained to output these two evidence parameters (α_c^i, β_c^i) for each input x^i. Following the relationship between Dirichlet and Beta distributions (where Beta is Dirichlet with C=2), and mapping to belief masses (Eq. (1)) and uncertainty (Eq. (2)), the belief masses for the positive (b_1^i) and negative (b_2^i) classes, and the uncertainty (u^i) become:

$$b_1^i = \frac{\alpha_c^i - 1}{\alpha_c^i + \beta_c^i}, \quad b_2^i = \frac{\beta_c^i - 1}{\alpha_c^i + \beta_c^i} \tag{6}$$

$$u^i = \frac{2}{\alpha_c^i + \beta_c^i} \tag{7}$$

Here, b_1^i represents belief in the event occurring, b_2^i represents belief in it not occurring, and u^i quantifies the lack of total evidence (higher $\alpha + \beta$ means lower uncertainty). The prediction is based on the expected probability $E[p_c^i] = \alpha_c^i/(\alpha_c^i + \beta_c^i)$.

One-versus-all Classifiers for Multi-Event WED. To handle multiple events ($C > 1$) efficiently, we employ a one-versus-all (OVA) strategy combined with parameter sharing (detailed in Section 5). Instead of C separate models, we use a shared feature extraction backbone. For each event c, a separate small classification 'head' is attached to the backbone. This head is specifically trained as a binary EDL classifier (outputting α_c^i and β_c^i) to distinguish event c from all other conditions (non-event c). Training involves optimizing C independent binary EDL loss functions, one for each event head, using appropriately labeled binary datasets derived from the original data (c vs. non-c). This allows the shared backbone to learn common representations while each head specializes, significantly reducing memory compared to deploying C full models.

The model learns mapping functions $h_c(x^i; \theta_{shared}, \theta_{head_c})$ where θ_{shared} are shared weights and θ_{head_c} are head-specific weights, such that:

$$(\alpha_c^i, \beta_c^i) = h_c(x^i; \theta_{shared}, \theta_{head_c}) \tag{8}$$

As mentioned, the final activation in each head must ensure positivity of α_c^i and β_c^i. We utilize ReLU layers due to their availability and efficiency in TFLM, ensuring non-negative outputs which serve as the evidence values.

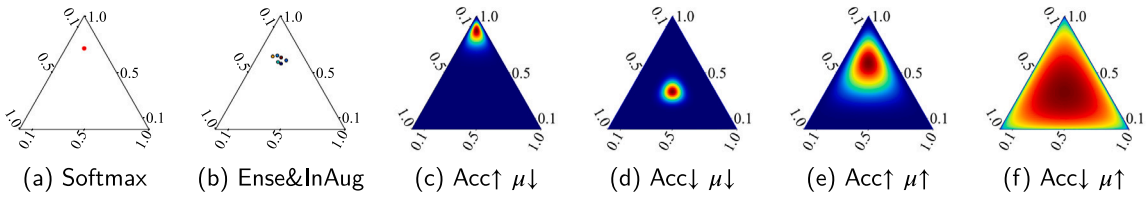


Fig. 3. Uncertainty estimation methods. (a) softmax model. (b) deep ensemble (Ense) and data augmentation (InAug). (c)–(f) EDL.

4.3. Uncertainty-aware training loss

To train the binary EDL classifiers within the OVA framework, we adapt the EDL loss principle. For each event c, we aim to minimize a loss function based on the predicted Beta distribution parameters $\psi_c^i = (\alpha_c^i, \beta_c^i)$ and the true binary label $y_c^i \in \{0, 1\}$. We utilize the Type II Maximum Likelihood loss from EDL theory, adapted for the Beta distribution output:

$$\mathcal{L}(\psi_c^i, y_c^i) = y_c^i(log(\alpha_c^i + \beta_c^i) - log(\alpha_c^i)) + (1 - y_c^i)(log(\alpha_c^i + \beta_c^i) - log(\beta_c^i)) \tag{9}$$

This loss encourages the model to produce high evidence (α_c^i or β_c^i) corresponding to the correct class label. Optionally, a regularizer based on the variance or entropy of the Beta distribution could be added to further refine uncertainty calibration, although Eq. (9) forms the core objective. We optimize the sum of these losses across all C event heads jointly [27], allowing the shared backbone and individual heads to learn effectively in the multi-task OVA setting.

Figs. 3(a) to 3(b) depict current uncertainty estimation methods available for MCUs, which include typical deterministic models (denoted as Softmax) , deep ensembles [28] (denoted as Ense), and data augmentation during test time [29] (denoted as InAug). Specifically, Fig. 3(a) shows a traditional deterministic model [30] that presents a point prediction and therefore has poor uncertainty estimation performance. In this figure, a three-class categorical probability of $[0.1, 0.1, 0.8]$ is mapped onto a simplex, with the left, top, and right vertices representing the three classes, respectively. Fig. 3(b) displays ensemble and data augmentation methods that require multiple inferences, resulting in a discrete simulation of the uncertainty.

In comparison, the evidential distribution (Figs. 3(c) to 3(f)) can provide both accurate classification (denoted as Acc) and uncertainty estimation (denoted as μ). Specifically, when the model is confident in its prediction (i.e., low uncertainty $\mu\downarrow$), a sharp distribution should be yielded on the simplex (Figs. 3(c) to 3(d)). Conversely, when the model is less confident (i.e., high uncertainty $\mu\uparrow$), the distribution becomes more diffuse (Figs. 3(e) to 3(f)).

5. Cascade learning architecture

This section details the design of the neural network architecture employed by the proposed framework, focusing on efficiency through shared representations and early exits. We first discuss the motivation and implementation of intra-event sharing for single-event detection, followed by the inter-event sharing strategy for handling multiple concurrent events, and conclude with the training methodology. All computationally intensive architecture search and model training procedures are performed offline on a server prior to MCU deployment.

5.1. Intra-event sharing via uncertainty-aware early exits

Deep learning models often exhibit redundancy, where shallower layers suffice to classify 'easy' input samples accurately, while deeper layers are only necessary for more complex or ambiguous inputs [31]. Exploiting this observation through early-exit mechanisms can significantly reduce computational cost. However, implementing efficient model sharing and dynamic inference paths on MCUs presents unique challenges due to severe constraints on processing power, memory availability, and supported software operations compared to more powerful edge devices like GPUs.

Nested Architecture with Early Exits. To enable intra-event sharing, we propose a nested cascade architecture comprising three sequential sub-networks: shallow, medium, and deep (visualized conceptually in Fig. 2). These sub-networks are constructed using repeating blocks of identical structure, drawing inspiration from established efficient architectures like MobileNetV2 [32], adapted for MCU constraints. Crucially, unlike traditional early-exit approaches that often rely on prediction confidence (e.g., max softmax probability) as the exit criterion, the proposed framework leverages the uncertainty estimate provided by the EDL framework (Section 4.2). We posit that model uncertainty is a more direct and reliable indicator of when a prediction is trustworthy enough to allow an early exit. Therefore, at the output of the shallow and medium sub-networks, the calculated uncertainty u^i (Eq. (7)) is compared against a pre-defined threshold ϵ. If the uncertainty is below the threshold ($u^i \leq \epsilon$), indicating a confident prediction, inference terminates, and the output from that stage is used. Otherwise, computation proceeds to the next, deeper stage. This allows the system to dynamically allocate computational effort, saving significant MCU resources on easier samples.

Architecture Search for Efficient Backbones. Selecting an appropriate backbone architecture is critical for minimizing MCU overhead. Building on findings that the number of operations (OPS) and network channel sizes heavily influence performance on MCUs [14], we implement a targeted Neural Architecture Search (NAS) to identify efficient structures for the three cascade stages.

Algorithm 1: Offline Search and Training Procedure for the Proposed Framework

Input: Channel search space L, OPS search space O, D^{TRAIN}
Output: Trained cascade model weights $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2$
Data: Training data D^{TRAIN}
/* Architecture Search (Simplified) */
1 best_backbone_config, best_score = None, 0
2 **for** i *in* L **do**
3 **for** j *in* O **do**
  // Train candidate backbone ($\mathbf{b}_{ij}$) config
4 NN_config ← Config(L_i, O_j)
  // Evaluate candidate (e.g., based on deep stage accuracy/OPS)
5 accuracy, ops ← Evaluate(NN_config, D^{TRAIN})
6 tradeoff ← accuracy/ops
7 **if** *tradeoff* > *best_score* **then**
8 best_backbone_config, best_score = NN_config, tradeoff
/* Cascade Training of Best Backbone */
9 **for** $l = 0, 1, 2$ *(Shallow, Medium, Deep)* **do**
  // Train stage l using output of stage $l-1$ (if $l > 0$)
10 Train stage $\mathbf{b}_l(\mathbf{W}_l)$ on relevant data subset
  // Stopping criterion (e.g., epochs, validation loss)
11 **if** *converged* **then**
12 continue
13 **return** $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2$

We employ Depthwise Separable Convolution blocks as the fundamental building unit, as these are known to be efficient proxies for latency control on MCUs [14]. Each block follows a standard pattern: a 1×1 point-wise convolution for channel mixing/expansion, followed by a 3×3 depthwise convolution for spatial feature extraction, and another 1×1 point-wise convolution for channel reduction/projection. The initial layer uses standard 2D convolutions for initial feature mapping from potentially varied input types. Consistent padding strategies ensure dimensional compatibility between blocks. A lightweight linear classification layer is attached after each stage (shallow, medium, deep) to produce the event prediction and EDL evidence parameters.

The search space explores channel sizes L from 32 to 512 and the number of Depthwise blocks per stage O from 3 to 7, values informed by successful MCU models like MobileNet [32], DSCNN [33], and MicroNets [14]. This defines a discrete search space of configurations. The objective is to find the configuration N^* that optimizes a trade-off between predictive accuracy (typically evaluated using the final, deep stage) and computational cost (proxied by OPS). Algorithm 1 conceptually outlines this process (Lines 1–9): iterating through configurations, evaluating them (often using proxy metrics during the search), and selecting the best-performing backbone structure according to the defined trade-off metric.

5.2. Inter-event sharing via multi-head architecture

For applications requiring the detection of C distinct event types from the same sensor input, naively deploying C independent models is often infeasible on MCUs due to memory and computational constraints [13]. However, different event types derived from the same sensor stream (e.g., different keywords in audio, different activities from motion sensors, different heart conditions from ECG) might share relevant underlying features. Exploiting this potential for shared representation learning is key to multi-event efficiency.

Shared Backbone with Multiple Heads. We propose a multi-event architecture where all C event detectors share the same optimized cascade backbone (shallow, medium, and deep stages identified via architecture search). Instead of separate models, we attach C independent classification 'heads' at each of the three potential exit points (shallow, medium, deep), resulting in $3 \times C$ heads in total (illustrated in Fig. 2). Each head corresponds to a specific event type and functions as a binary EDL classifier as described in Section 4.2. This multi-head design provides flexibility, allowing detection of any combination of events based on the application needs, while maximizing parameter reuse through the shared backbone. Compared to forcing a single multi-class output layer, this OVA approach with separate heads is often more suitable for WED where events might not be mutually exclusive and where individual event detector performance is critical. Each head typically consists of an adaptive pooling layer (to handle potentially different feature map sizes from the three stages) followed by a linear layer that outputs the α and β evidence parameters for its specific event. All heads and the shared backbone are trained jointly using a multi-task learning objective, optimizing the sum of the individual binary EDL losses (Eq. (9)) across all C events.

Uncertainty-aware Cascade Training Procedure. Training the complete cascade model (with shared backbone and multiple heads) efficiently requires careful consideration. Inspired by deep cascade learning principles, we adopt a stage-wise training approach, conceptually outlined in Algorithm 1 (Lines 10–14). We first train the parameters of the shallow stage (including its C associated heads). Then, freezing the shallow stage parameters, we train the medium stage (and its heads), potentially using

the outputs or intermediate representations from the shallow stage. Finally, the deep stage (and its heads) is trained, utilizing representations from the medium stage. This staged training helps stabilize the learning process for deeper networks and allows each stage to build upon the features learned by the preceding ones. While Algorithm 1 shows a simplified view, practical implementations might involve joint fine-tuning phases or different optimization schedules for each stage. Throughout training, each head produces both predictions and uncertainty estimates based on the EDL formulation.

The design leverages capabilities of modern TinyML libraries like TFLM, specifically its support for multi-tenancy execution (allowing different model components/heads to share memory resources efficiently) and optimized memory planners that facilitate reusing activation buffers, further reducing the runtime overhead compared to managing truly independent models.

6. System implementation and optimization

This section describes the hardware platforms, software toolchains, and specific optimization techniques employed to deploy and evaluate the proposed framework effectively on MCUs.

6.1. Hardware and software environment

Hardware Platforms. Model training and initial development were conducted on a Linux server utilizing an Intel Xeon Gold 5218 CPU and an NVIDIA Quadro RTX 8000 GPU. For on-device deployment and evaluation, we selected two representative MCUs commonly found in wearable and embedded systems: (1) the STM32F446ZE (denoted F446ZE), featuring an ARM Cortex-M4 processor, 128 KB SRAM, and 512 KB eFlash memory; and (2) the STM32H747XI (denoted H747XI), which includes a dual-core ARM Cortex-M7/M4 processor, 1 MB SRAM, and 2 MB eFlash. To maintain focus on typical single-core MCU scenarios, our evaluations on the H747XI utilized only the Cortex-M7 core, effectively limiting available resources to 512 KB SRAM and 1 MB eFlash.

Software Toolchain. The offline training pipeline for the proposed framework and baseline models was developed using PyTorch 1.8. The core evidential uncertainty computations were implemented using Python with NumPy. For deploying models onto the target MCUs, we adopted the TFLM framework [26], chosen for its widespread adoption, portability, and support for optimized kernels. The deployment code, including any online logic, was written in C++. Model conversion from PyTorch to the TFLM format involved an intermediate step using the ONNX standard representation followed by the TF Lite converter tool. The converted models were executed on the MCUs using the TFLM interpreter running on top of the Mbed OS real-time operating system. On-device signal pre-processing (e.g., MFCC feature extraction for audio) utilized the CMSIS-DSP library, while efficient neural network computations leveraged the optimized CMSIS-NN kernels integrated within TFLM.

6.2. Deployment strategies and optimizations

Multi-tenancy for Efficient Resource Sharing. To efficiently manage the multiple components of the cascade architecture (shallow, medium, deep stages, and multiple heads) within the tight memory constraints of MCUs, we leverage TFLM's capabilities for multi-tenancy deployment. This approach allows multiple logical model interpreters (representing different components or even different baselines like ensemble members) to share memory resources from a common pool dynamically allocated by TFLM's memory planner. This contrasts with statically allocating separate memory regions for each component, thereby significantly reducing the overall SRAM footprint, especially for models with shared backbones or ensemble methods. This strategy was applied consistently across the proposed framework and all relevant baselines during evaluation.

Uncertainty Operator Implementation with Standard Ops. A key practical challenge is calculating the EDL uncertainty (Eq. (7)) on the MCU using only standard, readily available TFLM operators. The calculation $u^i = 2/(\alpha_c^i + \beta_c^i)$ requires obtaining the evidence values α_c^i and β_c^i (outputs of a network layer), ensuring their positivity, summing them, and performing a division. We achieve this using a sequence of standard TFLM operations: (1) A ReLU activation is applied to the network layer outputs producing the evidence values, ensuring $\alpha_c^i, \beta_c^i \geq 0$ (as required by EDL and sufficient given the loss function). (2) The two evidence values are added using a standard *add* operator. (3) The constant value 2 is divided by the sum using a standard *divide* operator. This sequence avoids custom operators, enhancing portability. Although TFLM supports a *reduce_sum* operator conceptually, its kernel might not always be included in minimal builds; therefore, relying on element-wise *add* is often more robust. These operator sequences are integrated directly into the exported TFLM model graph (visualized in Fig. 4(a)), minimizing runtime overhead for uncertainty calculation.

MCU Library Footprint Optimization. Beyond model size, the memory footprint of the ML library itself (TFLM interpreter, kernels, OS) is critical on MCUs, which lack the large off-chip memory of mobile devices (cf. Fig. 1). Initial analysis revealed that even a quantized shallow model required substantial memory (e.g., 79 KB SRAM, 203 KB eFlash on F446ZE, including TFLM runtime, OS, etc.), potentially exceeding the budget of lower-end MCUs (like the STM32F205VB with 64KB SRAM). To mitigate this, we performed targeted optimizations on the TFLM library build itself. First, we identified and removed unused operator kernels and associated source files not required by our specific model architectures (Depthwise Conv, Pointwise Conv, ReLU, Add, Pool, Reshape, Divide). Second, we analyzed the linker map and potentially reordered object files to improve code locality, although the primary savings came from removing unused code. As shown in Fig. 4(b), this optimization significantly reduced the TFLM runtime overhead (e.g., from 122 KB to 32 KB in eFlash for the analyzed case), making deployment feasible even on highly constrained devices. This library optimization was applied uniformly to all methods evaluated.

Note on Quantization Impact: We employ standard 8-bit post-training integer quantization as a common technique to reduce model size and potentially improve latency. Across all evaluated methods, we observed only minimal impact (maximum 1% drop)

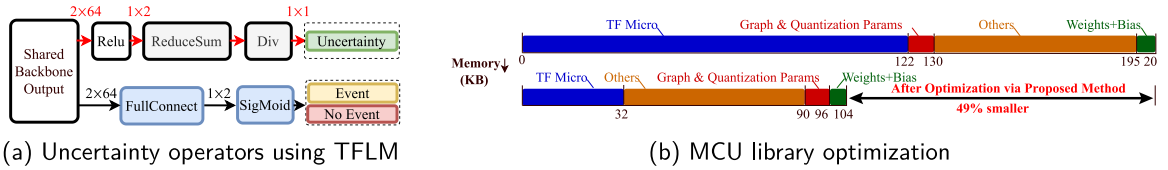


Fig. 4. Deployment stage details. (a) Sequence of standard TFLM operators used for on-device uncertainty calculation. (b) Comparison of MCU library eFlash footprint before (top) and after (bottom) targeted optimization by removing unused components.

on classification accuracy due to quantization. A detailed analysis of quantization effects specifically on uncertainty calibration metrics is considered future work.

Uniform Application of Optimizations: It is important to emphasize that the deployment strategies (multi-tenancy) and optimizations (library footprint reduction, quantization) were applied consistently to both the proposed framework and all baseline methods during the comparative evaluations to ensure fair assessment of the core algorithmic benefits.

7. Evaluation settings

This section outlines the experimental setup used to evaluate the proposed framework, including the datasets, performance metrics, and baseline methods employed for comparison.

7.1. Evaluated datasets

Our evaluation targets WED scenarios common in mobile health and human activity recognition. We selected four publicly available datasets representing different sensing modalities and event types relevant to the proposed framework's intended applications:

In-ear Audio Activity Recognition (Oesense). The Oesense dataset [34] comprises in-ear audio recordings from 31 subjects performing five activities: "walk", "run", "still", "drink", and "chew". For pre-processing, audio signals were segmented into one-second windows with a 4 kHz sampling rate. We extracted 10 Mel-frequency cepstral coefficients (MFCCs) using an 80 ms frame length and 40 ms stride, resulting in a $1 \times 10 \times 21$ input tensor per segment. The dataset was split into 90% for training (40,064 samples) and 10% for testing (4452 samples) across all activities.

Audio Keyword Spotting (KWS). We used the Google Speech Commands V2 dataset (KWS V2) [35], containing 105,829 one-second utterances from 2618 speakers across 35 words. Following common practice, we defined 12 classes: ten specific keywords, one 'silence' class, and one 'unknown' class encompassing the remaining words. Audio clips were processed at a 16 kHz sampling rate. We extracted 10 MFCCs using 40 ms frames, 20 ms stride, and 640 FFT points, yielding a $1 \times 10 \times 51$ input tensor. The split resulted in 92,502 training samples (90%) and 10,278 test samples (10%).

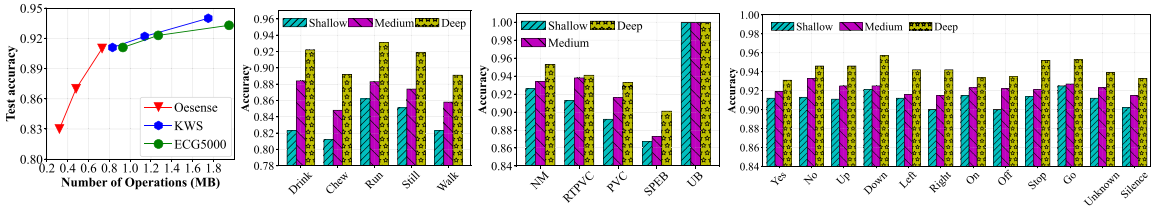
ECG Heartbeat Classification (ECG5000). The ECG5000 dataset [36] provides 20 h of single-channel ECG recordings, annotated into 92,584 individual heartbeats belonging to five classes: Normal (NM, 58.4%), R-on-T Premature Ventricular Contraction (RTPVC, 35.3%), Premature Ventricular Contraction (PVC, 3.9%), Supra-ventricular Premature or Ectopic Beat (SPEB, 2%), and Unclassified Beat (UB, 0.5%). Each heartbeat originally consists of 140 samples; we resampled these to 560 samples and reshaped them into a $1 \times 10 \times 56$ input tensor to match the 2D convolution input format used for consistency. This dataset exhibits significant class imbalance. The data was split into 4500 training samples (90%) and 500 test samples (10%). Notably, the UB class has only one sample in the test set, limiting robust evaluation for this specific rare event.

Motion Dataset (HHAR). The HHAR dataset [37] contains smartphone and smartwatch accelerometer and gyroscope data from 9 participants performing 6 activities (biking, sitting, standing, walking, stairs up/down). Collected using diverse hardware (8 smartphones, 4 smartwatches) and sampling rates (resampled to 100 Hz), it tests robustness. We used 6-channel smartwatch inertial data (accelerometer+gyroscope), segmented into 5-second windows (input: $1 \times 500 \times 6$). The split yielded approx. 2916 training (90%) and 324 test (10%) samples.

For all datasets exhibiting significant class imbalance (particularly ECG5000 and potentially KWS depending on keyword frequency), we applied SMOTE [38] exclusively to the *training* data. Recognizing that WED tasks often suffer from significant class imbalance where events of interest are rare, we specifically apply SMOTE during the training phase. Compared to simple random oversampling, SMOTE generates synthetic minority samples in feature space, potentially leading to better generalization and reduced overfitting. Unlike random undersampling, it avoids discarding potentially useful information from the majority (non-event) class. While more advanced techniques like ADASYN [39] or specialized ensemble methods [40] exist, SMOTE provides a well-established and effective balance between performance in handling imbalance and implementation simplicity, making it a suitable choice for pre-processing the training data before applying our primary EDL-based model.

7.2. Uncertainty evaluation metrics

To assess the quality of the uncertainty estimates produced by the proposed framework and the baselines, we employ three standard metrics commonly used in the uncertainty quantification literature:



(a) Model sizes *vs.* Acc. (b) Acc. on Oesense (c) Acc. on ECG5000 (d) Acc. on KWS

Fig. 5. Model sizes *vs.* accuracy and early exit performance results for single events across the evaluated datasets. Note the extremely small test set size for the Unclassified Beat (UB) event in ECG5000.

- **Brier Score:** Measures the mean squared difference between predicted probabilities and actual outcomes. Lower scores indicate better calibration and accuracy.
- **Negative Log-Likelihood (NLL):** Evaluates how well the predicted distribution fits the observed data. Lower NLL suggests the model assigns higher likelihood to the true outcomes.
- **Expected Calibration Error (ECE):** Quantifies the discrepancy between predicted confidence (e.g., max probability or belief) and empirical accuracy across different confidence bins. Lower ECE indicates better calibration, meaning the model's confidence aligns well with its actual accuracy.

7.3. Uncertainty quantification baselines

We compare the proposed framework against representative baseline approaches for uncertainty quantification, selected based on their relevance, performance characteristics (accuracy vs. efficiency), and potential deployability on MCUs:

- **Softmax Baseline:** A standard deterministic neural network with the same backbone architecture as the proposed framework but using a standard softmax output layer, representing typical non-uncertainty-aware models.
- **Vanilla EDL** [24]: This baseline uses the same backbone architecture but applies the standard EDL formulation (as described in Section 4.1) without the cascade or early-exit mechanisms proposed in the proposed framework. It serves to isolate the benefits of our architectural contributions beyond basic EDL. It is considered a state-of-the-art (SOTA) approach for *efficient* single-pass uncertainty estimation.
- **Deep Ensembles (D(Softmax)+Ense)** [7]: Widely regarded as a SOTA method for *accurate* uncertainty quantification. This baseline involves training an ensemble of N identical deterministic softmax models (using the same backbone) with different random weight initializations. Predictions and uncertainties are derived by averaging the outputs across the N models. We use $N = 5$, a common choice balancing performance and computational cost [41]. Despite its accuracy, ensembling significantly increases computational and memory demands.
- **Test-Time Data Augmentation (D(Softmax)+InAug)** [42]: A more *memory-efficient* alternative to ensembles for estimating uncertainty. It involves performing multiple inference passes (K) on slightly augmented versions of each test sample using a single deterministic softmax model. Uncertainty is estimated from the variance of predictions across augmentations. We use $K = 5$ augmentations, applying Jittering (adding Gaussian noise with mean $\varepsilon = 0$, std dev $\sigma = 0.03$) to the input features.

Note that Monte Carlo Dropout (MCDP) [11], another popular uncertainty technique, was not included as a baseline primarily because standard TFLM implementations typically do not support enabling dropout layers during inference, which is necessary for MCDP. Furthermore, its computational profile is often similar to or higher than deep ensembles, while reported uncertainty performance can be lower [10].

8. Results and analysis

This section presents the empirical evaluation of the proposed framework against the baselines defined in Section 7.3 and investigate the impact of key design choices, such as the cascade architecture and uncertainty thresholds.

8.1. Event detection and uncertainty performance

We first evaluate the fundamental trade-off between model complexity (proxied by OPS and parameters) and accuracy achieved by the searched cascade models, as well as the overall uncertainty performance compared to baselines. Networks were trained using the Adam optimizer (learning rate 1e−3, batch size 32) with early stopping based on validation loss (patience of 5 epochs).

Fig. 5(a) (left panel) shows the relationship between model size (parameters) and average accuracy across all events for the first three datasets, considering the shallow, medium, and deep stages of the searched architecture used in the proposed framework. As expected, accuracy generally improves with model depth/size. However, the gains diminish, particularly for KWS and ECG5000. For instance, on Oesense, moving from the medium (0.87 Acc, 0.58 MB) to the deep (0.91 Acc, 0.76 MB) stage yields a modest 4%

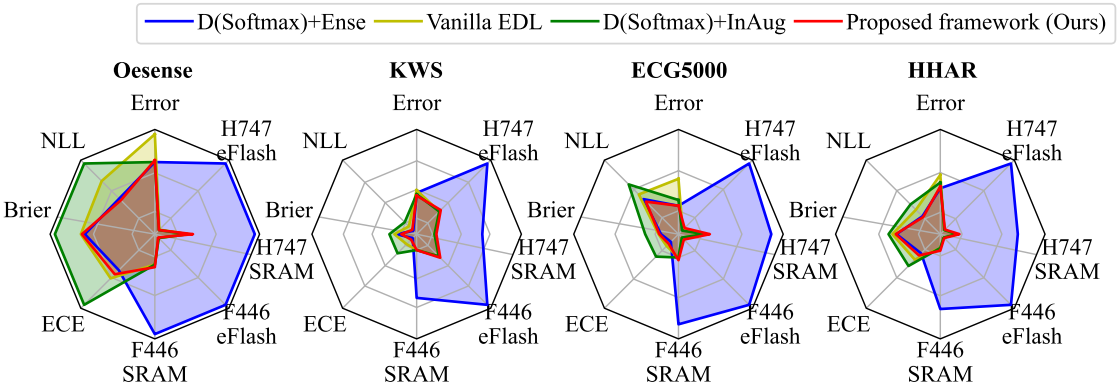


Fig. 6. Comparing Vanilla EDL, data augmentation, deep ensembles (SOTA), and the proposed framework using uncertainty, error rate, and memory usage metrics, averaged across four datasets (Oesense, KWS, ECG5000, HHAR). Lower values are preferred for all metrics.

accuracy improvement but requires a 31% increase in parameters. Similarly, for ECG5000, achieving a 1% accuracy gain necessitates roughly doubling the model size from shallow to deep. This highlights the potential for significant resource savings via early exits, as even the shallow models achieve respectable accuracy (>80% on average) with minimal footprints. Figs. 5(b) to 5(d) provide per-event accuracy details for the different cascade stages across the datasets.

Fig. 6 provides a comparative overview of the proposed framework against Vanilla EDL, Data Augmentation (D(Softmax)+InAug), and Deep Ensembles (D(Softmax)+Ense) using key uncertainty metrics (Brier, NLL - lower is better), error rate (1-Accuracy), and memory usage (eFlash, SRAM) on both MCUs, averaged across all four datasets. The proposed framework demonstrates superior uncertainty calibration compared to the Data Augmentation baseline across all datasets, achieving up to 22% lower NLL scores, indicating better-fitting predictive distributions. Compared to the highly accurate but resource-intensive Deep Ensembles, the proposed framework achieves comparable or slightly better performance on the uncertainty metrics (e.g., 8.0% lower Brier score on KWS, 1.7–2.4% lower NLL on Oesense/ECG5000) and similar error rates. Crucially, the proposed framework also significantly outperforms Vanilla EDL (which lacks the cascade structure) on these metrics, underscoring the benefit of the architectural contributions. Specifically for the HHAR motion dataset, the proposed framework achieved 92.5% accuracy, comparable to Deep Ensembles (92.4%) but with strong uncertainty scores (NLL 44.8, Brier 51.6) and minimal resource usage (43-44KB SRAM, 145-148KB eFlash), further highlighting its suitability for diverse WED tasks on MCUs. The radar plot visually summarizes how the proposed framework achieves a compelling balance, matching or exceeding the uncertainty performance of strong baselines while drastically reducing resource requirements (detailed further in Section 8.4).

Notably, these results underscore that the proposed framework achieves SOTA-level uncertainty performance while operating within a significantly reduced resource envelope compared to methods like Deep Ensembles (elaborated in Section 8.4), validating its efficiency for MCU deployment without sacrificing reliability.

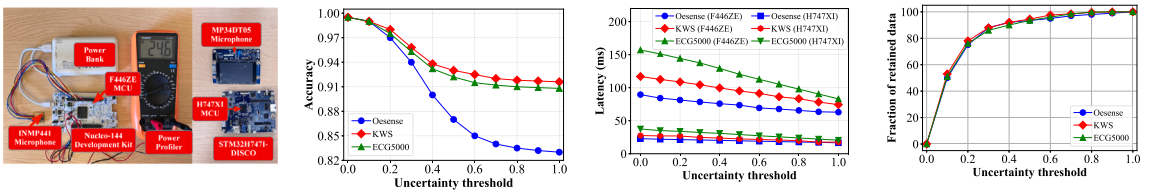
8.2. Impact of uncertainty thresholds on early exits

The early-exit mechanism in the proposed framework relies on a user-defined uncertainty threshold, ϵ. This section analyzes the impact of varying ϵ on system behavior, providing a more detailed interpretation of the results shown in Figs. 7(b), 7(c), and 7(d).

Fig. 7(b) illustrates the inherent trade-off between accuracy and computational savings mediated by the threshold. As ϵ increases, allowing samples with higher uncertainty to exit early, the overall accuracy tends to decrease. This is expected, as more challenging samples that might benefit from deeper processing are instead classified by shallower, potentially less accurate stages. The rate of accuracy degradation varies across datasets, likely reflecting differences in the proportion of 'easy' versus 'hard' samples. Conversely, setting a very low ϵ forces nearly all samples through the full deep model, maximizing potential accuracy but forfeiting the efficiency gains of early exiting. This configurability is a key feature, allowing practitioners to tailor the system's behavior. For instance, critical healthcare applications (e.g., detecting severe cardiac events in ECG5000) would likely demand a very low ϵ to prioritize reliability, ensuring only highly confident predictions are made by earlier stages. Whereas applications like simple activity or keyword detection might tolerate a higher ϵ to conserve battery life.

The direct impact on computational efficiency is clearly shown in Fig. 7(c). Average inference latency decreases monotonically as ϵ increases for all datasets on both MCU platforms. This reduction can be substantial, demonstrating the effectiveness of the uncertainty-based filtering. The corresponding plot in Fig. 7(d) quantifies the underlying mechanism: higher thresholds lead to significantly lower retention rates, meaning a larger fraction of samples successfully exit at the shallow or medium stages. For example, moderate threshold values might allow 50%–70% of samples to exit early, roughly halving the average number of layers processed per inference.

In essence, the uncertainty threshold acts as a crucial control knob, enabling a configurable balance between the desired level of predictive reliability (reflected in accuracy) and the operational efficiency (latency and associated energy consumption). This adaptability enhances the practical applicability of the proposed framework across diverse WED scenarios with varying constraints and requirements.



(a) End-to-end deployment (b) Accuracy vs. uncertainty threshold ϵ (c) Latency vs. uncertainty threshold ϵ (d) Retention rate vs. uncertainty threshold ϵ

Fig. 7. Setup and impact of uncertainty thresholds on the end-to-end system.

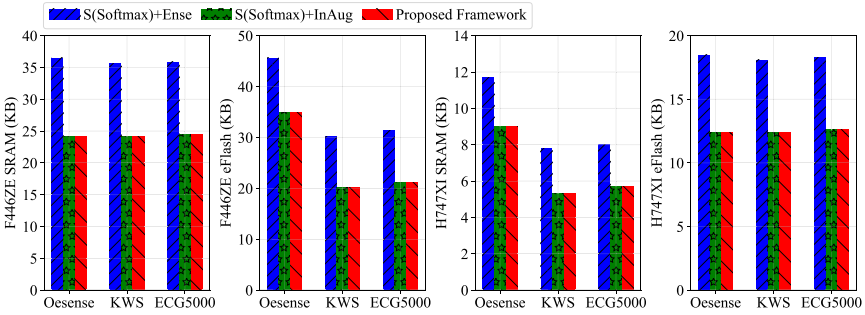


Fig. 8. Shallow models' SRAM and eFlash memory usage on two MCUs for three datasets between the proposed framework and baselines.

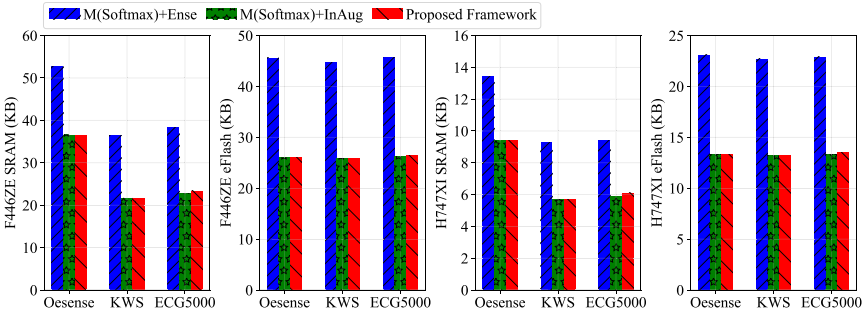


Fig. 9. Medium models' SRAM and eFlash memory usage on two MCUs for three datasets between the proposed framework and baselines.

8.3. Impact of memory usage on model sizes

The results obtained from the memory usage analysis reveal distinct trends and numerical differences across shallow, medium, and deep model configurations. For shallow models (Fig. 8), the SRAM of the F446ZE shows memory usage values of 36.5 KB for the shallow model via ensemble, 24.2 KB for the medium model via data augmentation, and 24.2 KB for the proposed framework. Similar trends are observed in the eFlash of the F446ZE, the SRAM of the H747XI, and the eFlash of the H747XI, where the proposed framework consistently demonstrates lower memory usage.

For medium models (Fig. 9), the SRAM of the F446ZE reports 52.8 KB for the medium model via ensemble methods, 36.6 KB for the medium model via data augmentation, and 23.4 KB for the proposed framework, indicating a significant reduction in memory consumption by the proposed framework. Similar patterns are observed in the eFlash of the F446ZE, the SRAM of the H747XI, and the eFlash of the H747XI.

In the case of deep models (Fig. 10), the SRAM of the F446ZE shows a pronounced difference with 75 KB for the deep model via deep ensemble, 48 KB for the medium model via data augmentation, and 49 KB for the proposed framework. The eFlash of the F446ZE further amplifies this observation, with the proposed framework maintaining lower memory usage compared to other methods. Similar patterns are noted in the SRAM and eFlash of the H747XI.

Overall, the general trend indicates that the proposed framework consistently outperforms the deep ensemble in terms of memory efficiency, demonstrating its robustness and effectiveness across various model configurations and scales.

Notably, the proposed framework achieves memory usage similar to data augmentation, which only deploys a singular model, but the proposed framework offers better uncertainty performance compared to data augmentation (cf. Section 8.1).

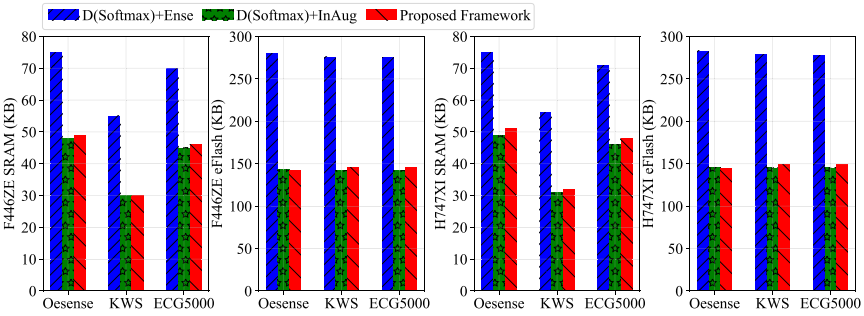


Fig. 10. Deep models' SRAM and eFlash memory usage on two MCUs for three datasets between the proposed framework and baselines.

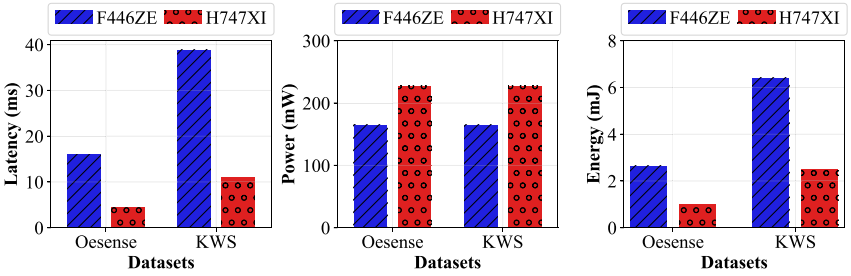


Fig. 11. MFCC feature extraction overheads. Raw signals are applied for ECG5000 in testing.

8.4. End-to-end system efficiency

Following the optimization of all baselines and the proposed framework using techniques including multi-tenancy deployment, model quantization, and MCU library optimization, we evaluate their runtime efficiency during deployment on MCUs (Fig. 7(a)). Our evaluation encompasses the entire system, including signal acquisition, feature extraction, and memory usage in terms of SRAM and eFlash required for model execution. We conducted experiments with various datasets and two typical resource-constrained MCUs, the F446ZE and H747XI. Although the focus is primarily on the ECG5000 dataset due to page limits, note that consistent outcomes were observed across all four datasets.

Model Inference Memory Footprint. Based on our implementation, the proposed framework consumes only 49 KB and 51 KB of SRAM (38.5% and 9.9% of the total SRAM of F446ZE and H747XI, respectively) as shown in Fig. 6. Additionally, as shown in Fig. 6, the proposed framework requires 142 KB and 145 KB of eFlash (27.7% and 14.1% of the total eFlash of F446ZE and H747XI, respectively). These results demonstrate that the proposed framework consumes only a small portion of the limited resources of MCUs, leaving enough resources for other applications to be supported simultaneously. Furthermore, the proposed framework requires only 66%–67% of SRAM (49 KB vs. 75 KB for F446ZE and 51 KB vs. 75 KB for H747XI) and 51% of eFlash (142 KB vs. 280 KB for F446ZE and 145 KB vs. 283 KB for H747XI) compared to the deep ensembles baseline.

Signal Acquisition Overheads. To evaluate signal acquisition overheads for the F446ZE MCU, we employ an INMP441 MEMS microphone. For the H747XI, we use the MP34DT05-A built-in microphone on the H747I-DISCO evaluation board (Fig. 7(a)). We assess energy consumption and memory usage as key factors. Energy consumption (J) is computed as the product of time/latency (t) and power (W). Power is determined from input voltage (V) and current measurements (A), conducted with a Fluke 87V digital multimeter. For the F446ZE, we record a power consumption of 24.6 mA at 3.3 V, resulting in 81.18 mJ for one second of audio signal acquisition. Memory-wise, it uses 4KB of SRAM and 32KB of eflash. In contrast, the H747XI consumes 31.6 mA at 3.3 V, totaling 104.28 mW in power. It utilizes 29KB of SRAM and 66KB of eflash. Overall, signal acquisition overheads for these two MCUs are minimal.

Feature Extraction Overheads. As shown in Fig. 11, the proposed framework demonstrates varying performance metrics for the Oesense and KWS datasets, measured across two MCUs, F446ZE and H747XI. For the Oesense dataset, which involves the MFCC feature extraction overhead, the system exhibits latency values of 16.033 ms and 4.505 ms for the F446ZE and H747XI MCUs, respectively. In contrast, for the KWS dataset, the latency values are 38.909 ms and 10.913 ms for the same MCUs. The power consumption for both the Oesense and KWS datasets is recorded as 165 mW for the F446ZE and 227.7 mW for the H747XI. Regarding energy consumption, the Oesense dataset shows values of 2.645 mJ and 1.026 mJ for the F446ZE and H747XI MCUs, respectively, whereas the KWS dataset reports energy values of 6.420 mJ and 2.485 mJ for the corresponding MCUs. Additionally, raw signals are applied for the ECG5000 dataset in testing.

These metrics indicate several important implications for the performance of the MCUs. The H747XI MCU consistently achieves lower latency compared to the F446ZE MCU. For instance, a latency of 38.909 ms on the F446ZE for the KWS dataset means that

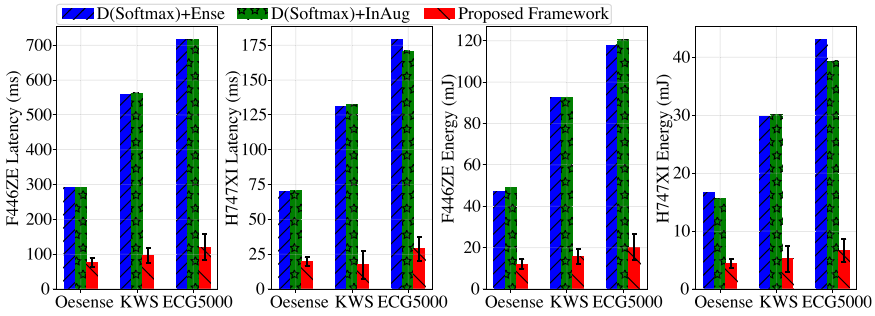


Fig. 12. Comparison of latency and energy consumption of uncertainty-aware methods on two MCUs.

each operation takes nearly 39 ms to complete. This is significant for real-time event detection applications such as keyword spotting, which typically have a duration of about one second. The lower latency of 10.913 ms on the H747XI means it can process events much faster on higher-end MCUs, enhancing the system's responsiveness.

While the H747XI consumes more power (227.7 mW) compared to the F446ZE (165 mW), this needs to be balanced with the performance benefits. Higher power consumption can be justified if it leads to significantly better performance metrics such as lower latency and energy efficiency. Energy consumption is a critical factor as it combines both power usage and the time taken to complete tasks. For example, the energy consumption of 6.420 mJ for the KWS dataset on the F446ZE means that if the MCU were to continuously process events, it would consume a significant amount of energy over time.

In practical terms, if a battery-powered device has a capacity of 100 mAh at 1.5 V (which equals 150 mWh), the F446ZE could theoretically support approximately 23,364 events feature extractions (150 mWh/6.420 mJ) before the battery is depleted. In contrast, the H747XI's energy consumption of 2.485 mJ for the same dataset means it could support around 60,362 events, making it much more efficient for prolonged use.

Model Inference Latency. Using the MBed Timer API to measure latency on MCUs, Fig. 12 illustrates the proposed framework's and baseline inference results across three datasets and two MCUs. With uncertainty thresholds (u) ranging from 1 to 0, the proposed framework presents latencies from lowest to highest, respectively. While baseline approaches, like deep ensemble, yield reliable uncertainty estimations, they exhibit high inference latencies of 717.2–717.4 ms on F446ZE and 171.1–179.3 ms on H747XI per sample. Conversely, the proposed framework ensures both reliable uncertainty and minimized latency, cutting inference latencies up to 8.64× (83.0 ms vs. 717.2 ms) on F446ZE and 8.35× (20.2 ms vs. 171.1 ms) on H747XI. Moreover, the proposed framework enhances latency by approximately 4.56× against other baselines, even without uncertainty filtering.

Model Inference Energy Consumption. Similar to the latency results, the proposed framework significantly reduces energy consumption compared to the baselines, as shown in Fig. 12. For example, the proposed framework decreases energy consumption by up to 834% (116.0 mJ vs. 13.9 mJ) on F446ZE and 857% (39.4 mJ vs. 4.6 mJ) on H747XI when compared to the best-performing benchmark uncertainty-aware baselines. Also, we observe that the proposed framework achieves around 450% energy improvement compared to the baselines without uncertainty filtering.

In summary, the above-discussed metrics highlight that the overhead and memory usage of the proposed framework are small, ensuring that the system remains efficient and reliable across different datasets and MCUs. This efficiency makes it particularly suitable for real-time applications and battery-powered devices, supporting a high number of events before battery depletion.

8.5. Qualitative robustness to signal corruptions

Beyond standard performance metrics, the practical value of uncertainty estimation lies partly in its ability to signal potential issues with the input data or model limitations. To illustrate this, we qualitatively examined the proposed framework's behavior when presented with corrupted input signals, specifically segments with missing data (zeroed out) and segments with added Gaussian noise. Fig. 13 contrasts the proposed framework's output with that of a standard softmax model (same architecture) on a "Chew" event sample. The softmax model may yield confident but incorrect predictions on the corrupted inputs. The proposed framework, however, demonstrates more desirable behavior: it often maintains the correct classification despite the corruption, and critically, when the input degradation causes misclassification or ambiguity, the associated uncertainty output by the proposed framework increases markedly (u approaches 1.0). This heightened uncertainty effectively flags the prediction as unreliable due to the input anomaly. While not a quantitative analysis, this example highlights the potential of the proposed framework's integrated uncertainty estimate to improve overall system robustness by identifying conditions under which the model's output should not be trusted.

9. Discussion

Generalizing the proposed framework to other sensors and higher-end MCUs. Ideally, the proposed framework could be generalized to any wearable sensors driven by MCUs. However, sensor signal complexity and limited MCU memory size pose

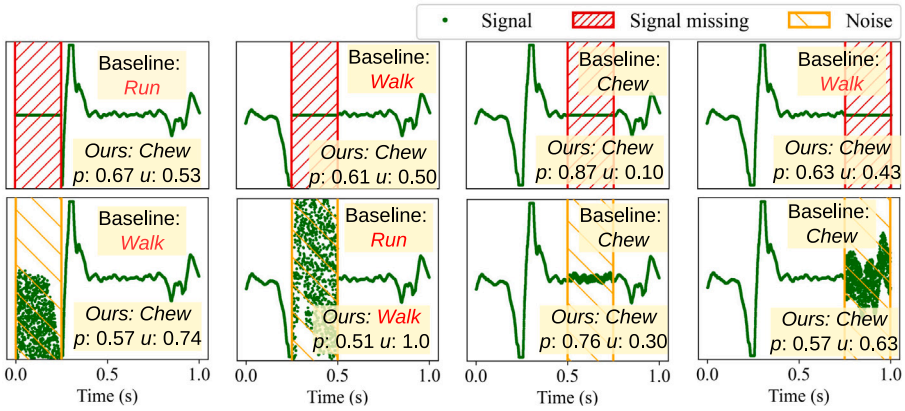


Fig. 13. Illustrative example of uncertainty estimation behavior under signal corruption (missing data, noise). Red labels indicate incorrect predictions. The proposed framework (noted as "Ours") often shows high uncertainty (u) for corrupted inputs, especially when misclassifying.

limitations. More complex signals usually require larger model sizes, challenging the deployment on the constrained memory of MCUs. Fortunately, recent work [13] shows that by investigating compressive sensing, key patterns of primitives in signals can be compressed and extracted, which indicates it can reduce the model size to save system overhead. Therefore, we will study how compressive sensing combined with the proposed framework could further reduce system overhead to generalize to ultra low-end MCUs. We envision our method could also benefit higher-end MCUs, e.g. STM32F4, which has 1MB flash and 192KB SRAM. Since less memory is required, higher-end MCUs could experience improvements in latency and energy efficiency.

Impact of the proposed framework on future WED systems. Our work has illustrated that uncertainty is a key criterion to ensure reliable prediction in WED systems. Therefore, an important and urgent question is how to define uncertainty tolerance thresholds for specific applications. Fortunately, for healthcare applications, we can design this criterion through a doctor-in-the-loop strategy to select the optimal threshold.

Limitations. Despite the advancements presented, this work has limitations. (1) *Hardware/Software Dependence:* The implementation relies on specific STM32 MCUs and the TFLM library. Porting to different hardware or frameworks like uTVM might require re-implementing optimizations or finding alternative operator sequences for uncertainty calculation. (2) *Evaluation Scope:* While evaluated on four diverse datasets covering audio, ECG, and inertial motion, its performance on other potentially relevant sensor modalities, such as photoplethysmography (PPG) common in wearables, or on more complex event types has not been explored. The effectiveness of the proposed architecture search and cascade learning strategy might also vary depending on the specific characteristics and complexity of different datasets and tasks. (3) *NAS Constraints:* The architecture search explored a defined space; more advanced NAS methods could find better architectures but require more computational resources for the search itself. The fixed three-stage design is also a simplification. (4) *Quantization Effects on UQ:* We focused on accuracy impact post-quantization; a deeper analysis of how 8-bit (or lower-bit) quantization affects the *calibration* and quality of the EDL uncertainty estimates themselves is needed. (5) *Threshold Tuning:* The reliance on manual tuning for the early-exit threshold ϵ limits ease of use and adaptability.

Future Directions. Building upon this work, future research could focus on: (1) Enhancing portability through more abstracted operator implementations or automated code generation for diverse targets. (2) Expanding the evaluation scope to include a wider variety of sensor modalities (e.g., PPG, Galvanic Skin Response (GSR), Electrooculography (EOG), Electromyography (EMG)) and more complex, potentially overlapping, event detection scenarios to further assess the framework's robustness and scalability. (3) Investigating advanced, MCU-aware NAS techniques for optimizing uncertainty-aware cascade models. (4) Performing rigorous analysis of quantization effects on uncertainty calibration and developing quantization-aware training for EDL models. (5) Designing adaptive or learnable mechanisms for dynamic uncertainty threshold adjustment. (6) Incorporating online learning or model adaptation capabilities to handle data drift in long-term deployments.

10. Conclusion

This paper presented a framework designed to enable efficient and reliable WED directly on resource-constrained microcontrollers. Extending significantly upon our preliminary work [8], this journal version provides a more comprehensive exposition of the proposed framework, which integrates Evidential Deep Learning (EDL) for single-pass uncertainty quantification with a novel cascade architecture optimized for MCUs. We detailed the architecture's use of uncertainty-driven early exits (intra-event sharing) and multi-head feature sharing (inter-event sharing) for efficiency and scalability. Furthermore, we elaborated on practical implementation aspects, including architecture search, 8-bit quantization, TFLM operator sequences for uncertainty calculation, and library optimization. Our expanded empirical validation across four diverse datasets representing different sensor modalities (audio, ECG, and inertial motion) and two MCU platforms demonstrates the framework's effectiveness. Compared to strong baselines like Deep Ensembles, the proposed framework achieves competitive accuracy and superior uncertainty calibration (e.g., up to 22% lower

NLL vs. data augmentation), coupled with dramatic resource reductions: up to 8.64× faster inference, 8.57× lower energy use, and up to 55% memory savings. This work confirms the potential of the proposed approach to significantly advance reliable, real-time, multi-event detection on ubiquitous low-power wearable devices.

CRediT authorship contribution statement

Hong Jia: Writing – original draft, Validation, Data curation, Visualization, Investigation, Conceptualization. **Young D. Kwon:** Software, Conceptualization, Writing – review & editing. **Dong Ma:** Writing – review & editing, Conceptualization. **Nhat Pham:** Conceptualization, Writing – review & editing. **Lorena Qendro:** Writing – review & editing, Conceptualization. **Tam Vu:** Conceptualization, Writing – review & editing. **Cecilia Mascolo:** Funding acquisition, Conceptualization, Investigation, Supervision, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by ERC through Project 833296 (EAR), and Nokia Bell Labs, USA through a donation.

Data availability

Data will be made available on request.

References

[1] A. Alavi, G.K. Bogu, M. Wang, E.S. Rangan, A.W. Brooks, Q. Wang, E. Higgs, A. Celli, T. Mishra, A.A. Metwally, et al., Real-time alerting system for COVID-19 and other stress events using wearable data, Nature Med. 28 (1) (2022) 175–184.
[2] C. Holz, E.J. Wang, Glabella: Continuously sensing blood pressure behavior using an unobtrusive wearable device, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1 (3) (2017) http://dx.doi.org/10.1145/3132024.
[3] Y. Zhang, Z. Yang, Z. Zhang, P. Li, D. Cao, X. Liu, J. Zheng, Q. Yuan, J. Pan, Breathing disorder detection using wearable electrocardiogram and oxygen saturation, in: Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, 2018, pp. 313–314.
[4] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, S. Han, Mcunet: Tiny deep learning on iot devices, 2020, arXiv preprint arXiv:2007.10319.
[5] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, J. Snoek, Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, Adv. Neural Inf. Process. Syst. 32 (2019).
[6] G. Carneiro, L.Z.C.T. Pu, R. Singh, A. Burt, Deep learning uncertainty and confidence calibration for the five-class polyp classification from colonoscopy, Med. Image Anal. 62 (2020) 101653.
[7] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Adv. Neural Inf. Process. Syst. 30 (2017).
[8] H. Jia, Y.D. Kwon, D. Mat, N. Pham, L. Qendro, T. Vu, C. Mascolo, Ur2m: Uncertainty and resource-aware event detection on microcontrollers, in: 2024 IEEE International Conference on Pervasive Computing and Communications, PerCom, IEEE, 2024, pp. 1–10.
[9] Y. Wang, M. Gu, M. Zhou, X. Qian, Attention-based deep Bayesian counting for AI-augmented agriculture, in: Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, 2022, pp. 1109–1115.
[10] L. Qendro, J. Chauhan, A.G.C. Ramos, C. Mascolo, The benefit of the doubt: Uncertainty aware sensing for edge computing platforms, in: 2021 IEEE/ACM Symposium on Edge Computing, SEC, IEEE, 2021, pp. 214–227.
[11] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.
[12] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, B. Lakshminarayanan, Simple and principled uncertainty estimation with deterministic deep learning via distance awareness, Adv. Neural Inf. Process. Syst. 33 (2020) 7498–7512.
[13] N. Pham, H. Jia, M. Tran, T. Dinh, N. Bui, Y. Kwon, D. Ma, P. Nguyen, C. Mascolo, T. Vu, PROS: an efficient pattern-driven compressive sensing framework for low-power biopotential-based wearables with on-chip intelligence, in: Proceedings of the 28th Annual International Conference on Mobile Computing and Networking, 2022, pp. 661–675.
[14] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, P. Whatmough, Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers, Proc. Mach. Learn. Syst. 3 (2021) 517–532.
[15] E. Liberis, Ł. Dudziak, N.D. Lane, uNAS: Constrained neural architecture search for microcontrollers, in: Proceedings of the 1st Workshop on Machine Learning and Systems, EuroMLSys '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 70–79, http://dx.doi.org/10.1145/3437984.3458836.
[16] A. Ghodrati, B.E. Bejnordi, A. Habibian, Frameexit: Conditional early exiting for efficient video recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15608–15618.
[17] E. Bondareva, E.R. Hauksdóttir, C. Mascolo, Earables for detection of bruxism: a feasibility study, in: Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers, 2021, pp. 146–151.
[18] D. Wójcik, T. Rymarczyk, M. Oleszek, Ł. Maciura, P. Bednarczuk, Diagnosing cardiovascular diseases with machine learning on body surface potential mapping data, in: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, SenSys '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 379–381, http://dx.doi.org/10.1145/3485730.3492883.
[19] T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, J. Song, Occlumency: Privacy-preserving remote deep-learning inference using SGX, in: The 25th Annual International Conference on Mobile Computing and Networking, 2019, pp. 1–17.

[20] T. Chen, Y. Li, S. Tao, H. Lim, M. Sakashita, R. Zhang, F. Guimbretiere, C. Zhang, NeckFace: Continuously tracking full facial expressions on neck-mounted wearables, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5 (2) (2021) http://dx.doi.org/10.1145/3463511.
[21] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, H. Kwon, Clio: Enabling automatic compilation of deep learning pipelines across iot and cloud, in: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 2020, pp. 1–12.
[22] A. Malinin, M. Gales, Predictive uncertainty estimation via prior networks, Adv. Neural Inf. Process. Syst. 31 (2018).
[23] J. Mukhoti, A. Kirsch, J. van Amersfoort, P.H. Torr, Y. Gal, Deep deterministic uncertainty: A simple baseline, 2021, ArXiv E-Prints, arXiv–2102.
[24] M. Sensoy, L. Kaplan, M. Kandemir, Evidential deep learning to quantify classification uncertainty, 2018, arXiv preprint arXiv:1806.01768.
[25] A. Malinin, B. Mlodozeniec, M. Gales, Ensemble distribution distillation, 2019, arXiv preprint arXiv:1905.00076.
[26] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang, et al., Tensorflow lite micro: Embedded machine learning for tinyml systems, Proc. Mach. Learn. Syst. 3 (2021) 800–811.
[27] G. Franchi, A. Bursuc, E. Aldea, S. Dubuisson, I. Bloch, One versus all for deep neural network incertitude (OVNNI) quantification, 2020, arXiv preprint arXiv:2006.00954.
[28] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017, URL https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.
[29] M.S. Ayhan, P. Berens, Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks, in: Medical Imaging with Deep Learning, 2018.
[30] C. Malaviya, P. Ferreira, A.F. Martins, Sparse and constrained attention for neural machine translation, 2018, arXiv preprint arXiv:1805.08241.
[31] X. Dai, X. Kong, 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.
[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, 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.
[33] Y. Zhang, N. Suda, L. Lai, V. Chandra, Hello edge: Keyword spotting on microcontrollers, 2017, arXiv preprint arXiv:1711.07128.
[34] D. Ma, A. Ferlini, C. Mascolo, OESense: employing occlusion effect for in-ear human sensing, in: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, 2021, pp. 175–187.
[35] P. Warden, Speech commands: A dataset for limited-vocabulary speech recognition, 2018, arXiv preprint arXiv:1804.03209.
[36] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu, E. Keogh, A general framework for never-ending learning from time series streams, Data Min. Knowl. Discov. 29 (6) (2015) 1622–1664.
[37] A. Stisen, H. Blunck, S. Bhattacharya, T.S. Prentow, M.B. Kjærgaard, A. Dey, T. Sonne, M.M. Jensen, Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, 2015, pp. 127–140.
[38] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[39] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1322–1328.
[40] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C (Appl Rev) 42 (4) (2011) 463–484.
[41] L. Qendro, A. Campbell, P. Lio, C. Mascolo, Early exit ensembles for uncertainty quantification, in: Machine Learning for Health, PMLR, 2021, pp. 181–195.
[42] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, H. Xu, Time series data augmentation for deep learning: A survey, 2020, arXiv preprint arXiv:2002.12478.