SoundSieve: Seconds-Long Audio Event Recognition on Intermittently-Powered Systems

Mahathir Monjur, Yubo Luo, Zhenyu Wang, Shahriar Nirjon
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
{mahathir,yubo,zywang,nirjon}@cs.unc.edu

ABSTRACT

A fundamental problem of every intermittently-powered sensing system is that signals acquired by these systems over a longer period in time are also *intermittent*. As a consequence, these systems fail to capture parts of a longer-duration event that spans over multiple charge-discharge cycles of the capacitor that stores the harvested energy. From an application's perspective, this is viewed as sporadic bursts of missing values in the input data – which may not be recoverable using statistical interpolation or imputation methods. In this paper, we study this problem in the light of an intermittent audio classification system and design an end-to-end system – *SoundSieve* – that is capable of accurately classifying audio events that span multiple on-off cycles of the intermittent system. SoundSieve employs an offline *audio analyzer* that learns to identify and predict important segments of an audio clip that must be sampled to ensure accurate classification of the audio. At runtime, SoundSieve employs a lightweight, energy- and content-aware *audio sampler* that decides when the system should wake up to capture the next chunk of audio; and a lightweight, intermittence-aware *audio classifier* that performs imputation and on-device inference. Through extensive evaluations using popular audio datasets as well as real systems, we demonstrate that SoundSieve yields 5%–30% more accurate inference results than the state-of-the-art.

CCS CONCEPTS

• **Computer systems organization → Embedded software**.

KEYWORDS

Audio perforation, scheduling, sampling, classification

ACM Reference Format:
Mahathir Monjur, Yubo Luo, Zhenyu Wang, Shahriar Nirjon. 2023. SoundSieve: Seconds-Long Audio Event Recognition on Intermittently-Powered Systems. In *The 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3581791.3596859

MobiSys '23, June 18–22, 2023, Helsinki, Finland

ACM ISBN 979-8-4007-0110-8/23/06…$15.00
https://doi.org/10.1145/3581791.3596859

1 INTRODUCTION

As batteryless computing systems continue to mature, we see a gradual shift in research from developing tools and platforms for energy harvesting systems [12, 21, 22, 60, 67], to devising new programming paradigms and runtime systems [13, 14, 23, 29], to more recently, implementing machine learning techniques and applications [26, 28, 35, 47, 70] tailored to intermittently-powered systems that run on harvested energy from solar, kinetic, thermal, or RF sources. A wide variety of on-device inference-capable intermittent systems have been proposed in the recent literature [11, 17, 20, 27, 34, 35, 45, 46, 52, 53, 56, 68, 73] that perform on-device inference of audio, image, environmental parameters, building activity, and human activity recognition tasks.

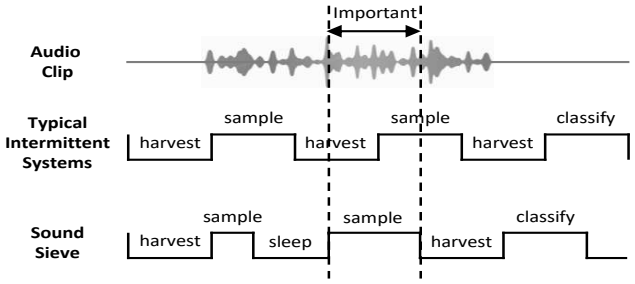


Figure 1: A typical intermittent system, for lack of content awareness, misses important portions of the audio. SoundSieve samples and analyzes an earlier segment, decides (in real-time) to conserve energy by sleeping (and also by harvesting while in sleep mode) and use that energy later to capture the important segment. Note that there can be multiple non-consecutive important segments in an audio. We only show one for illustration.

Unfortunately, existing intermittent systems have only been demonstrated successful in classifying short-lived events that last for about a few hundred milliseconds [52]. This limit is fundamental to the design of intermittent systems that require a fixed-sized capacitor to accumulate enough harvested energy for the system to wake up and remain active. The system in its active mode consumes the harvested energy to perform application-specific tasks, but soon runs out of energy and goes back to sleep. By choosing a larger (or smaller) capacitor, the lengths of the active and sleep phases can be increased (or decreased), but generally, there will always be one or more sleep phases when the system is unable to acquire any sensor data. This is why state-of-the-art intermittent systems implicitly assume that the event of interest lasts no more than one discharging cycle of the capacitor so that it can be captured fully in one active phase of the system. Events that span across multiple charge-discharge cycles of the capacitor are not fully captured by

existing intermittent systems. Hence, specialized techniques are required to ensure accurate classification of longer-duration events on intermittently-powered systems.

In this paper, we take up the challenge of audio event classification on intermittent systems where events may last longer than one charging cycle (and possibly more). To achieve this, we exploit two empirically-learned properties of audio signals: first, *relevance* – i.e., not every segment of an audio clip contributes equally to its classification; and second, *imputability* – i.e., many segments within an audio clip can be imputed or estimated to a sufficient detail from other nearby segments.

These observations lead us to the design of *SoundSieve* — a software-only, content- and energy-aware solution to seconds-long audio event classification on intermittent systems. The key idea behind SoundSieve is its ability to predict and control when an intermittent system should wake up from sleep – in order to capture and process only the *relevant, non-imputable* segments of the input audio. Unlike existing intermittent systems that continuously sample and process audio until the capacitor is fully drained of harvested energy, SoundSieve saves energy by switching to sleep mode when the incoming audio does not carry information that is new or relevant to classification. This strategy increases SoundSieve's discharge time since the capacitor now can hold charge and use that energy later. Because of this, SoundSieve is able to sense and process input signals during intervals when existing intermittent systems cannot. This contrast is illustrated in Figure 1.

Although the idea of SoundSieve is simple and intuitive, developing such a solution is challenging due to severe CPU, memory, and energy constraints of an intermittent system. SoundSieve cannot afford computationally expensive sampling techniques that are suitable for battery-powered, advanced mobile systems such as robotic explorers [18, 32, 75]. SoundSieve must sample a small segment of the input audio, analyze its content, and decide whether to continue sampling or to go to sleep (and if so, for how long). All of these must happen in real-time, and ideally, without any additional energy overhead. SoundSieve achieves this through a combination of offline and online algorithms.

SoundSieve employs an offline *audio analyzer* that learns to identify and predict important segments of an audio clip that must be sampled to ensure accurate classification of the audio clip. At runtime, SoundSieve employs a lightweight, energy- and content-aware *audio sampler* that decides when the system should wake up to capture the next audio segment. The sampler's decision algorithm runs in parallel with the signal acquisition process to ensure that there is no gap in signal acquisition. The decision algorithm purposefully applies a convolution filter on the sampled audio to make accurate sampling decisions based on high-level acoustic features. Since this filter is also the first layer of the classifier (by design), there is practically no extra energy overhead of the audio sampler. Once an audio event ends, a lightweight, intermittence-aware *audio classifier* performs imputation and classification.

Several salient features make SoundSieve unique of its kind. First, SoundSieve is the first intermittent system that makes content-aware sampling and processing decisions to enable long duration audio event recognition. Second, SoundSieve is a single-node, software-only solution that neither requires multiple coordinated harvesters, nor any modification to the harvester, nor any additional hardware components beyond a basic intermittent computing system. Third, SoundSieve is agnostic to the harvester and intermittence pattern. Even battery-powered systems can adopt SoundSieve's intelligent sensing mechanism to extend their battery-life. Fourth, SoundSieve's signal processing framework is generalizable beyond audio. It can inspire future intermittent systems that detect complex patterns in time-series signals [77, 78] such as video [41] and motion sensors [40, 42, 79].

In order to evaluate SoundSieve, we conduct dataset-driven experiments as well as real-world deployments with these systems. In the dataset-driven experiments, we compare the performance of SoundSieve against three baseline solutions, including state-of-the-art intermittent system [52], over four datasets that contain over 140,000 audio clips from over 125 categories of sounds having the duration of up to 4 seconds. We observe that SoundSieve outperforms all baselines by a significant margin — SoundSieve successfully captures 5%–30% more (relevant) audio segments, and as a result, achieves 5%–25% higher inference accuracy than the baseline solutions. For the real-world deployments, we develop a 16-bit TI MSP430FR5994-based intermittent system that senses audio in real-time and performs on-device inference while being powered by solar or RF energy. We implement two applications that involve recognizing voice commands and household activities, respectively. We demonstrate that SoundSieve captures and accurately classifies 20% more events than the baseline intermittent systems which do not perform content-aware sampling.

2 EMPIRICAL STUDY

In order to understand the effect of missing values caused by intermittence, we conduct an empirical study on three popular audio datasets: Urban8K [64], ESC-50 [61], and GSC [71]. These datasets contain over 115,000 audio clips of over 95 different categories.

2.1 Dealing With Missing Values

A classifier that does not explicitly handle missing values performs extremely poorly when it encounters such inputs. This is evident in Figure 2 – confirming similar prior studies [58] – where we see 20%–50% drop in inference accuracy (referred to as *do nothing*) when 75% of the data are missing.

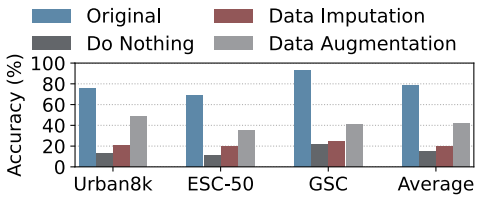


Figure 2: Imputation and augmentation increase accuracy, but cannot maximize the gain in accuracy by themselves. The percentage of missing values in the experiment is 75%.

To address this issue, there are two generic classes of solutions in the literature: (a) estimating (imputing) the missing values (before classification) using statistical or neural methods, and (b) training the classifier on an augmented dataset where missing values are synthetically introduced in the training examples. Although implementing any or both of these techniques improve the classification

accuracy – referred to as *imputation* and *augmentation*, respectively – there is still a large gap of 5%–40% in accuracy that cannot be regained. Furthermore, the classifier used in this experiment is a state-of-the-art CNN having 1008 filters, which is not feasible for intermittent systems. In conclusion, **data imputation and/or data augmentation cannot completely solve the intermittent audio classification problem. We need to complement them with audio content-aware intelligent sampling at the front to maximize their performance.**

2.2 Feasibility of Content-Aware Sampling

Although an intermittent system does not have complete control over the harvesting (charging) cycles, which depend on the energy source, once charged, the system can decide when to use that harvested energy. This creates an opportunity for the system to selectively sample segments of the audio that are more informative and relevant to audio classification.

We conduct another study to understand what percentage of an audio clip must be sampled so that we have enough information to classify a clip correctly. To conduct this study, we (logically) divide each clip into non-overlapping 100ms *segments* and search for the smallest set of K most informative segments that must be sampled to correctly classify the clip. We keep the K selected segments intact, and impute the rest of the segments prior to classifying the clip. The outcome of this experiment is summarized in Table 1.

Sound Category	#Clips	K=10%	K=30%	K=50%	K=70%	K=90%
Voice Command	600	64.20%	81.50%	88.70%	100.00%	100.00%
Vibration	120	56.62%	81.18%	92.31%	93.47%	94.20%
Alarm	85	54.62%	85.09%	91.91%	93.11%	93.50%
Children	80	56.62%	88.82%	92.19%	94.61%	95.95%
Music	76	48.52%	82.29%	89.61%	91.47%	94.14%
Pet	76	52.27%	83.61%	92.10%	94.27%	95.10%
Animal	65	54.68%	90.63%	92.21%	93.00%	100.00%
Bursts	16	51.13%	65.90%	81.81%	85.20%	92.00%
Laundry	16	54.10%	69.70%	83.10%	89.40%	98.00%
Sleep	16	56.21%	72.27%	80.53%	88.96%	100.00%
Wellness	16	64.32%	85.42%	88.78%	100.00%	100.00%
Bathroom	12	61.11%	83.33%	90.00%	100.00%	100.00%
Drink	12	55.32%	67.40%	84.18%	86.24%	95.00%
Overall	1190	51.47%	85.91%	89.60%	92.30%	96.60%

Table 1: The percentage of clips that are classified correctly when we sample *K%* most informative segments.

Each row of Table 1 is a cumulative distribution of K for a certain audio category, where K denotes the minimum number of most informative segments to sample to classify an audio correctly. For example, the last number of the first row tells us – if we sample the most informative 90% segments (and impute the remaining 10%), we can correctly classify 93.5% of the alarm sounds from Urban8k dataset. Figure 3 shows the same information as in Table 1 but separately for each dataset.

This result provides insights into the redundancy in audio clips for classification purposes. For an intermittent system, from its charge and discharge times, we can estimate what fraction of the time the system can sample, and then estimate their expected inference accuracy from these distributions. These datasets, however, are trimmed and preprocessed to remove unwanted sounds. In real-world audio, we expect more redundancy and thus an intermittent system is expected to have more room for selecting informative segments to achieve higher accuracy than the estimated

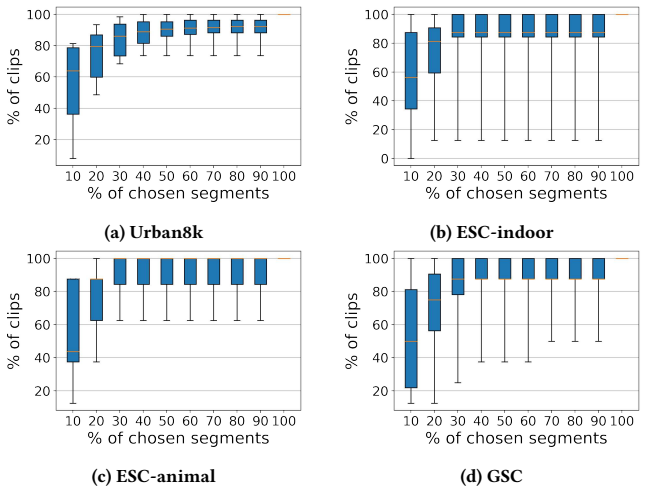


Figure 3: The distribution of correctly-classified clips in each dataset when K most informative segments are sampled.

values in this study. In conclusion, **not all audio segments being equally informative, a content-aware sampler on a resource-constrained sensing system can select a subset of informative audio segments to maximize their ability to correctly classify long-duration audio clips.**

2.3 Sparsity of Audio Events

Using a large capacitor for intermittent audio event classification ensures continuous energy supply, but can result in missed events if charging time is longer than gaps between consecutive events.

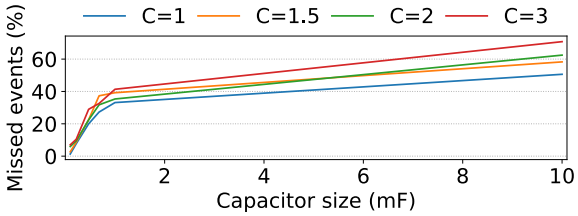
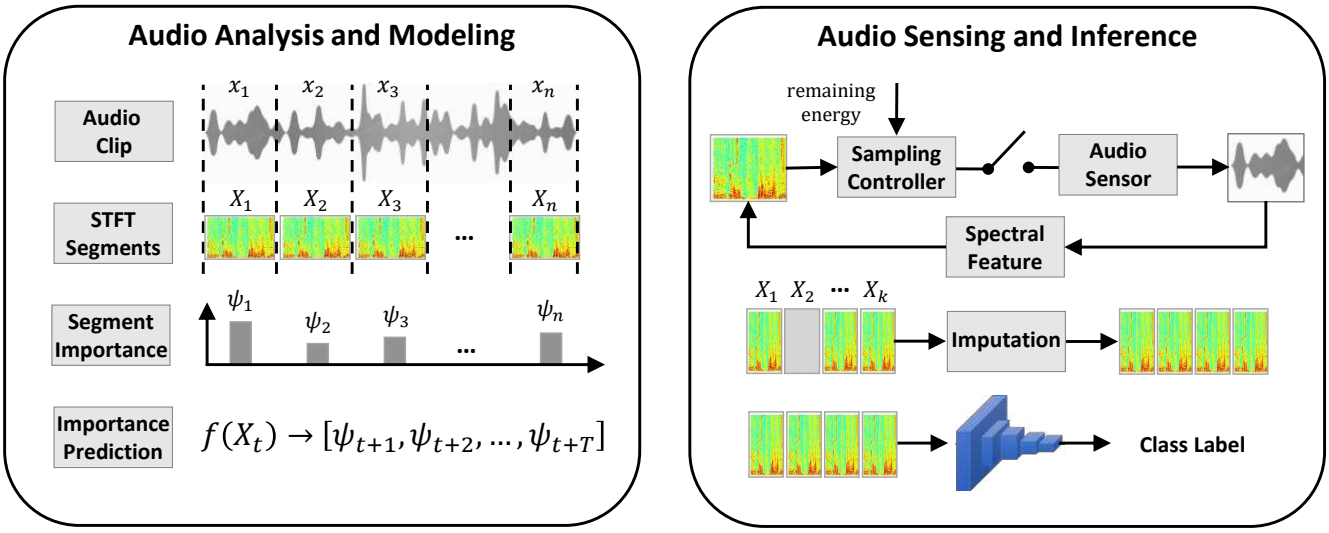


Figure 4: Percentage of audio events that are missed for different capacitor sizes at different energy harvesting rates. C denotes the ratio between charging and discharging times of the capacitor.

We conduct an experiment using the DCASE SELD dataset [62] to measure the percentage of missed audio events due to capacitor charging and discharging times in intermittent systems with varying capacitor sizes ($100\mu F$ to $10mF$) and energy harvesting parameters (1 to 3). The dataset contained 600 audio scenes, recorded in different environments with real-world background noise. The results are used to quantify what fraction of the audio events are missed (completely) by a typical intermittent system.

Figure 4 shows that with larger capacitors, e.g., $10mF$, as many as 70% of the audio events are missed in real-world scenarios. In contrast, smaller capacitors, e.g., $100\mu F$, results in as low as 4% missed audio events, but the captured audio events have many missing samples or holes in them – which calls for an intelligent audio sampling and processing technique that deals with missing samples in intermittent audio sensing systems.



(a) Offline Phase (b) Online Phase

Figure 5: SoundSieve consists of two phases: (a) an offline phase for audio analysis and modeling, and (b) an online (real-time) phase for audio sensing and inference on an intermittent system.

3 OVERVIEW OF SOUNDSIEVE

SoundSieve is an audio sensing and inference framework that enables seconds-long audio event recognition on intermittently-powered systems. At the heart of SoundSieve is a lightweight, content-aware sampling algorithm that decides (in real-time) when the system should wake up from sleep to actively sample the next few segments of the incoming audio. SoundSieve's sampling objective is to maximize the classification accuracy by capturing the most informative audio segments, given the remaining charge in the capacitor, which limits the maximum number of segments the system can sample before it runs out of harvested energy.

SoundSieve is a software-only solution which does not require specialized hardware or multiple collaborating nodes. The framework consists of two phases: (a) an offline phase for audio analysis and modeling, and (b) an online (real-time) phase for audio sensing and inference on an intermittent system. The offline phase involves processing a large dataset of audio recordings to extract features and train machine learning models for both audio event sampling and classification. The online phase involves deploying the trained models to an intermittent system, where the system can detect audio events in real-time. The online phase operates on a continuous cycle of energy harvesting, sensing, and classification. Figure 5 shows both phases along with the major processing steps in them.

3.1 Audio Analysis and Modeling

Audio Segment Importance. SoundSieve logically divides audio clips into non-overlapping 100ms *segments*, and processes one segment at a time. The segment size is empirically determined to optimize the system's ability to extract acoustic features under different energy harvesting conditions. Each audio segment is assigned an importance score that denotes how important that segment is for accurate classification of the clip. During the offline phase of SoundSieve, the importance score of each 100ms segment of every audio clip in the dataset is computed.

Predicting Segment Importance Scores. In order for SoundSieve to decide which segments to capture next, the system must be able to predict the importance of the next few segments in the incoming audio. To enable this, during the offline phase of SoundSieve, a simple regression network (which runs on the intermittent platform in real-time) is trained. The predictor predicts the importance score of up to 5 next segments, given the frequency domain features of the audio segment that has just been read.

3.2 Audio Sensing and Inference

Sampling. SoundSieve's audio processing system is triggered by the microphone, which remains in ultra-low-power listening mode and sends an interrupt to the microcontroller whenever a sound is detected. SoundSieve reads an audio segment and extracts its spectral features. The sampling controller uses these features and the offline-trained importance predictor to identify potential candidate segments to read, and based on the remaining charge in the energy buffer, it decides whether (and for how long) it should go into to the low-power mode before it samples the next segment.

Imputation. Once the audio event ends, prior to classifying, the missing segments in the sampled audio are imputed to estimate the missing values. A lightweight, hierarchical interpolation technique is used to impute the missing segments. Note that even though the missing segments are less important (according to the segment importance score predictor), SoundSieve still imputes them to form a continuous audio clip which can be fed to a neural network.

Classification. An offline-trained convolutional neural network (CNN) is used to classify the audio. The classifier is trained on an augmented dataset where missing values are synthetically introduced at random locations in the audio clip and then the audio is imputed. To be able to classify audio signals of arbitrary lengths, a global max pooling layer is used after the final convolution layer which sums out the spatial information, making it robust to the arbitrary spatial dimension of the input.

4 AUDIO ANALYSIS AND MODELING

During the offline phase, SoundSieve analyzes each audio clip in the dataset to estimate the importance of each segment and to generate a predictor that is able to predict the importance scores.

4.1 Audio Segment Importance

Since harvested energy is scarce, SoundSieve aims to sample only the highly informative segments to make the best use of the energy. Hence, there has to be an objective measure that quantifies the importance of each segment of an audio clip – which the sampling controller can use in its segment selection process.

Approach. A naive way to identify important segments of an audio clip is to remove that segment, estimate its value using the remaining segments, and then test if this modified audio clip can still be classified correctly by a classifier that correctly classifies the unmodified original clip. This technique, however, does not provide a score that quantifies the importance of the segment.

SoundSieve determines the importance of each segment of an audio clip by analyzing its relative contribution towards the classification of the clip. The idea is similar to feature selection problem when each segment is treated as a feature. A linear regressor takes these features as the input and tries to predict the output – which is a value as predicted by a baseline classifier (e.g., a neural network). The weights of the linear regressor indicates the relative importance of each feature (i.e. segment) when it tries to mimic the behavior of the baseline classifier.

Algorithmic Details. Given, a baseline classifier, f, that is trained on the entire dataset; and an audio clip, $\mathbf{x}$, consisting of n segments, following algorithmic steps are followed to quantify the importance of each segment:

Step 1 – Generate a large number of audio clips having missing values, $\mathbf{z}_i$, by randomly turning on and off arbitrary number of segments in $\mathbf{x}$.

Step 2 – Impute all missing values in $\mathbf{z}_i$ using a fast, lightweight imputation algorithm described later in this paper (Section 5.3).

Step 3 – Train a linear ridge regressor, g that minimizes a measure of how unfaithful g is in approximating f in the locality defined by a cosine distance metric. The loss function is as follows:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_i d(\mathbf{x}, \mathbf{z}_i)\left[f(\mathbf{z}_i) - g(\mathbf{z}_i; \mathbf{w})\right]^2$$

Here, $d(\mathbf{x}, \mathbf{z}_i)$ is a distance measure between the original clip, $\mathbf{x}$ and the augmented clip, $\mathbf{z}_i$; $\mathbf{w}$ is the learned weight vector whose elements $\{w_k\}$ are the desired importance scores of the segments in $\mathbf{x}$.

These three steps described above are repeated for each audio clip in the dataset.

4.2 Predicting Segment Importance Scores

Since the sampling controller in SoundSieve has to predict the importance of the next few segments that are yet to be sampled, it requires a predictor that can do so from the knowledge of recently sampled audio. As this predictor has to run on the intermittent system in real-time, it must be lightweight and low-overhead.

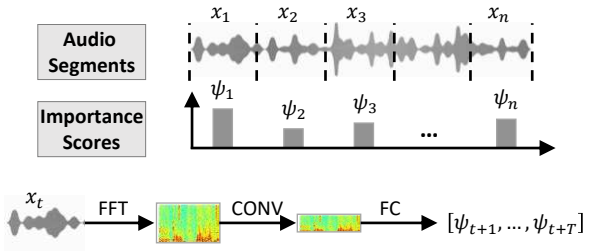


Figure 6: Segment Importance Predictor.

Approach. Given an observed audio segment at any instant, SoundSieve employs a lightweight neural network to predict the importance scores of the next few segments of the audio. The network is trained during the offline phase and used in the online (real-time) phase of SoundSieve.

The network consists of fully connected layers – which makes it fast and low-overhead. The network, however, takes spectral features of the audio segment as the input since they carry unique signatures of various types of audio events. Even though the predictor uses spectral features, these features are identical to the first layer of the audio classier (described in Section 5.4) which are reused. This is why they do not add extra overhead to the end-to-end audio processing cost in SoundSieve.

Algorithmic Details. The development and training process of the segment score predictor is as follows:

Step 1 – For each audio clip in the dataset, we randomly take a segment from the clip and take the importance scores of the next 5 segments. Depending on the length of the audio, the number of segments in a clip can be anything. SoundSieve's goal is to predict the next 5 segments' importance scores, given the current segment. To train this predictor, from each n-segment clip, $(n-6+1)$ training examples are generated.

Step 2 – Segments chosen in the previous step undergo a feature extraction process where a 1x4 convolution operation is applied after taking the short-time Fourier transform of the segment.

Step 3 – The regression neural network is trained using the spectral features as the input and the importance scores as the output.

Rationale behind Convolution. The convolution layer has a filter of dimension (1x4); hence, the filter only convolves along the frequency axis but not the time axis. The reason behind this design is three-fold. First, a filter that convolves along the time domain is not feasible since a segment is observed only for a very short period in time. Second, due to memory constraints, the dimensions of the sampled data needs to be lowered, which is achieved by convolution. Third, by using a convolution layer which overlaps with the audio classifier – we are able to extract fine-grained information for predicting the importance of next few segments.

4.3 Global Segment Importance

After computing the *local* importance of each segment for each audio clip, we aggregate the scores to compute a *global* importance score of the audio segments over the entire dataset. The process of computing global importance score of the audio segments is as follows:

Step 1 – The importance score of each segment of each clip are normalized to the range [-1, +1].

Step 2 – The average of all importance scores for each segment over all audio clips in the whole dataset is computed. This gives us a global importance score of each audio segment for the entire dataset.

Both global and local importance scores are used by SoundSieve to decide which segment to sample next. The rationale behind having a global importance score for each segment is that when none of the next few segments have significantly high local importance scores or there is a lot of uncertainty in the prediction of the local importance scores, but the system has energy to sample one or more segments, it needs another means to be able to rank the segments. This technique complements the local score-based sampling and improves SoundSieve's ability to sample informative segments.

5 AUDIO SENSING AND INFERENCE

During the online (real-time) phase, SoundSieve intelligently samples audio segments, imputes the missing values, and performs on-device inference.

5.1 Sensing and Preprocessing

SoundSieve remains in its ultra-low-power sleep mode when there is no sound in the environment and it has no pending tasks. This is enabled by the ultra-low-power listening mode of the microphone which continuously listens to the environment while being powered by harvested energy and sends an interrupt signal to the microcontroller only when audio activities are detected in the environment. This is when the microcontroller wakes up and switches to active mode (subject to availability of adequate harvested energy) to start its real-time audio sensing and inference processes.

SoundSieve samples audio data at the unit of 100ms segments. After reading an audio segment, it analyzes the segment (as part of the sampling controller algorithm that is described next) to decide whether to read the next segment or to go to sleep. This analysis process involves computing the spectral features of the audio segment that takes significant amount of time. Hence, SoundSieve samples and pre-processes a segment in parallel, so that sampling decisions can be made in real-time.

This is implemented by maintaining a four-element FIFO buffer queue, where each buffer holds 25ms of audio. The key idea is to pre-process data from a subset of the buffers in the low-energy accelerator (LEA) co-processor while the other buffer fills in with new data. The steps are as follows:

Step 1 – Initially, the first two buffers fill up (25ms + 25ms = 50ms).

Step 2 – While the spectral features are extracted on buffers 1 and 2, in parallel, the third buffer fills up.

Step 3 – While the spectral features are extracted on buffers 2 and 3, in parallel, the fourth buffer fills up.

Step 4 – The spectral features are extracted on buffers 3 and 4.

By using these buffers, we compute FFT of 50ms audio, which means the window length for the STFT is 50ms. We are also moving our window by one buffer during each iteration, meaning we are using a hop length of 25 ms. Additionally we are also using a convolution layer of shape (1x4), which means the convolution is done only in the frequency axis and not in the time axis. Hence it is not dependent on the previous or next temporal information.

5.2 Sampling Strategy

SoundSieve makes sampling decisions at runtime based on both the global as well as the local importance scores of the segments. The global importance scores are used to have an initial plan for segment selection and sampling. This initial sampling plan is modified as the system acquires new segments and gets more insights into the importance of the segments based on their local importance scores.

Parameter C. SoundSieve keeps track of its energy harvesting rate by computing the ratio, C between the charging and discharging times of the capacitor. This ratio characterizes the dynamics of an intermittent system. For example, a continuously-powered system has $C = 0$, while any intermittent system has $C > 0$. If $C = 1$, an intermittent system has to charge for the duration of exactly one segment in order to fully recover the energy it lost by sensing one segment. Therefore, the value of $C = 1.0, 1.5, 2.0, 3.0$ refers to 50%, 60%, 67%, 75% missing values in the data, respectively. Generally, $C = 1$ represents outdoor light (1200 lux), $C = 2$ represents indoor light (800 lux), and $C = 3$ represents dimly lit room (500 lux) and weak RF sources. Recent works on intermittent sensing have used a value of C within similar ranges. For example, [52] used $1 < C < 3$, and [7] used $0.8 < C < 1$ (outdoor solar), $1.1 < C < 3$ (indoor solar), and $1.0 < C < 3.2$ (indoor RF).

Initial Sampling Strategy. SoundSieve uses the parameter C to formulate an initial sampling strategy as follows:

Step 1 – At first the maximum number of segments, t_{max} that can be sampled is estimated from the current charging-to-discharging ratio, C. For instance, if $C = 2$, the system has to wait twice the amount amount of time in low-power sleep mode to recharge the capacitor to gain back the same amount of energy that is used for sampling and processing audio. This sets a limit on the number of segments that the system can sample and use in classifying the audio event.

Step 2 – A binary mask is created that tells the system which segment to sample and which to skip. Using both the charge-to-discharge ratio, C and the global segment importance scores, the mask values for the most important t_{max} segments are set to 1. These are the segments that the system should sense in order to maximize its classification performance.

Step 3 – In order to account for energy constraints, the mask values (i.e., the current sampling strategy) are checked to ensure that in no point in time the stored energy is completely exhausted, and if so, the mask value for that position is set to 0. This is a situation when the system must stop sampling for lack of energy even though the segment is important.

Adapting the Initial Sampling Strategy. SoundSieve relies on the low-power listening mode of the microphone to remain in low-power mode waiting for an audio event to occur. Once the microphone interrupts SoundSieve, the system transitions from low-power mode to active mode, and starts sampling and processing audio segments of length, $t_{segment}$.

The system starts with the initial sampling strategy, but the plan changes as it encounters new audio segments. The process is as follows:

Step 1 – STFT of the audio segment is computed and a 1x4 convolution is performed on the frequency axis to extract the frequency domain features.

Step 2 – The segment importance predictor is used to predict the importance of the next n segments. There are two cases: first, if the importance scores of any segment is higher than an empirically obtained threshold, or if the system is sitting idle with full charge, it decides to sample that segment that is locally important but may not have been included in the initial sampling plan. Second, if none of the importance scores is higher than the threshold, the system follows the initial sampling plan as the default.

After deciding which segment to sample next, unless it is the very next segment, SoundSieve switches to low-power sleep mode and wakes itself up when that segment arrives. This process is repeated until the end of audio event.

5.3 Imputation

Although SoundSieve's intelligent sampling algorithm has control over which segments to sample, it cannot avoid missing segments in the captured audio. Prior to classifying an audio clip that has missing values, SoundSieve estimates the missing values using a frequency domain imputation technique.

Approach. Prior to imputation, SoundSieve transforms time domain audio signals into frequency domain signals using the fast Fourier transform (FFT). The rationale behind imputing signals in the frequency domain is that audio events are characterized by their frequency components and the unique characteristics of different types of audio events are easily observable in the frequency domain. Furthermore, the frequency components of an audio event typically do not change drastically within a short period. Hence, interpolating the signal in the frequency domain is much more effective than doing so in the time domain.

Algorithmic Details. Given a sequence of audio segments where some of the segments are null (missing valued), following steps are performed to estimate the missing values:

Step 1 – Suppose, there are missing segments from time step, t_s to time step, t_e. Hence, the length of the missing segments is $t_s - t_e + 1$. The imputation process starts from these two ends, t_s and t_e.

Step 2 – At each time step, t, the frequency domain components are interpolated using following equations:

$$X(t,f) = \big[1 - r(t)\big]X(t_s - 1, f) + r(t)X(t_e + 1, f) \tag{1}$$

$$r(t) = \frac{t - (t_s - 1)}{(t_e + 1) - (t_s - 1)} \tag{2}$$

Step 3 – The previous step is repeated after the value of t_s has been increased and the value of t_e has been decreased, until $t_s < t_e$.

5.4 Inference

SoundSieve employs a convolutional neural network (CNN) to classify the audio event. The CNN consists of a combination of convolution, max pooling, and fully connected layers. The network architecture is shown in Table 2.

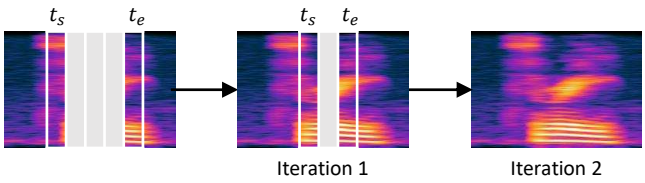


Figure 7: Illustration of imputation: during the first iteration, audio segments at time t_s and t_e are interpolated, leaving only one segment left for imputation. The process is repeated until all segments are imputed.

#	Layer	Dimensions	Parameters
1	Conv2D	60x32x1	5
2	Conv2D	29x30x2	20
3	Conv2D	12x13x8	152
4	Conv2D	4x4x32	2336
5	Dense	256	8448
6	Dense	10	2056

Table 2: Neural network architecture (13,017 parameters).

Approach. The inference process starts when sampling the audio event has completed as determined by a period of silence by the microphone in its ultra-low-power listening mode.

After sampling has completed and the missing audio segments have been imputed, the resultant short-time Fourier transform (STFT) values are converted to Mel-frequency spectrogram to reduce the dimension of the input signal. The spectrogram passes through to a sequence of convolution and max pooling layers. After the last convolution layer, a global max pooling layer is used — which takes the maximum value across time and frequency domains for each channel, and outputs one value for each of the channels. This design makes it possible for the model to handle audio signals of arbitrary lengths. Since audio events may have arbitrary lengths, a global max pooling layer is used to handle the variable-length input, instead of zero-padding. This makes the classifier model robust to audio events of arbitrary length.

The network is trained offline using both original audio clips as well as augmented audio clips where missing values are randomly introduced. The augmented audio clips are imputed using the method described in Section 5.3, before using them for training the network. This is done to ensure that the model is robust to missing audio segments.

6 EVALUATION ON DATASETS

6.1 Experimental Setup

Dataset. We use two popular audio classification datasets that contain audio clips of several seconds containing environmental sounds. We also use two datasets containing short speech commands and phrases. We divided these datasets into multiple sub-classses depending on the type of the sound. For example, ESC-50 [61] dataset covers a large variety of sound types such as animal sound, human activity, indoor noises etc. A brief summary of the datasets used in our study is shown in table 3. The duration of these clips are 1–5 seconds. Since we want to demonstrate SoundSieve's ability to classify multi-second audio, we collected some of our own data by recording as well as from online sources. Furthermore, we divided the dataset according to the sound types.

Dataset	Clips	Classes	Examples
Urbank8k [64]	8,732	10	Human, nature, mechanical.
ESC-50 [61]	2,000	50	Human activity, animal, indoor
GSC [71]	105,829	35	Speech commands.
Fluent Speech [44]	23,132	31	Short phrases.

Table 3: Datasets used in our study.

Modeling Intermittence. We use two parameters to model intermittence during our evaluation. First parameter is B which denotes the maximum number of audio segments the system can process without power failure. In our experiments all the audio segments have a length of 100ms. We also use another parameter C to denote the ratio between charging and discharging time to process one audio segment of 100ms. In this way, we discretize the energy pattern of an intermittent system. We start with an initial budget of B and each time we read and process audio segment of 100ms, we decrease the value of B. Additionally, whenever the system skips an audio segment and waits in low-power mode, we increase the energy budget by $\frac{1}{C}$. However since C can be floating point value, we consider only the lower bound while checking for available energy budget. Whenever the budget becomes zero, the system can not sense and process an audio segment, even if the importance of that segment is high.

Baseline Solutions. We use three baseline solutions for comparison: Vanilla, Periodic, and CIS [10]. In the vanilla approach, we emulate how an intermittent system normally works, i.e., sensing until the energy buffer is completely depleted and then waiting for it to be filled again. The periodic sampler samples every nth audio segment, where n depends on the value of C. CIS [10] uses multiple sensor nodes to capture and classify 283ms audio. We implement this approach by sensing 3 consecutive segments ($3 \times 100 = 300$ms) and then waiting for the energy buffer to be full again. We denote this baseline as CIS1 in our experiments.

Evaluation Metrics. We use inference accuracy to quantify the performance of the classifier. Accuracy refers to the portion of correctly classified instances on a dataset. We use detection of sound events for environmental sound classification dataset such as UrbanSound8k and ESC-50, and detection of certain phrases for GSC and Fluent Speech dataset. We also divide these dataset into multiple categories depending on the types of sound.

6.2 Comparison with the State-of-the-Art

In Figure 8, we compare SoundSieve's performance with that of state of the art approaches when the energy harvesting pattern is modeled by setting $C = 1$. We also show the original accuracy of the classifier when there is no missing audio segment at all. Although the classification accuracy drops when there are missing audio segments due to power failure in an intermittent system, we see that conserving energy to sense when there are more information available improves the overall classification accuracy by 3%-17% than other approaches.

Effectiveness of Importance Predictor. In order to demonstrate why SoundSieve can produce better classification accuracy than other approaches, we show the average coverage of the most informative segments in audio clips using both SoundSieve and vanilla approach in Figure 9. We use the term recall – the ratio between

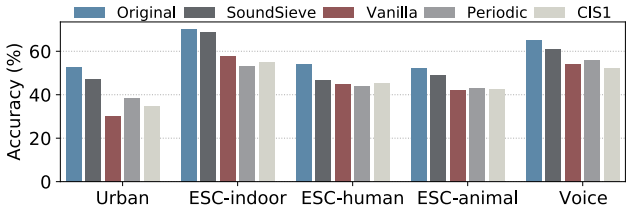


Figure 8: Comparison among different sampling approaches.

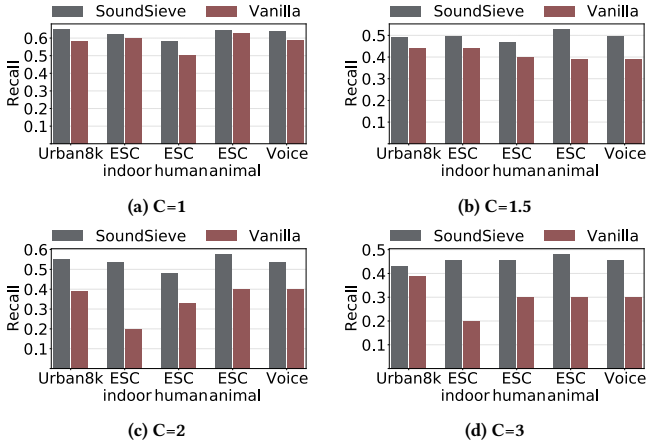


Figure 9: Comparison of coverage of the most informative audio segments at different energy harvesting rate

the number of positive samples correctly labeled as positive to the total number of positive samples.

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{3}$$

Here, a positive sample refers to an informative segment whereas a negative sample refers to a segment of lower importance. We use the true importance scores of each segment of an audio clip and the energy harvesting pattern to compute the most optimal sampling strategy and use that to measure the recall.

The values of recall clearly demonstrate that SoundSieve can sense as much as 25% more important audio segments compared to the vanilla approach. As we decrease the energy harvesting rate by setting the value of C higher, we observe greater improvement in covering only the most informative blocks compared to the vanilla approach.

6.3 Effect of Energy Harvester

We change the energy harvesting pattern in our simulation by varying the value of C. A higher value of C refers to a larger amount time required to harvest the energy of sensing one audio segment. It also means that a higher percentage of audio is missing in the audio clip. Figure 10 demonstrates that as we decrease the energy harvesting rate, the classification accuracy drops slightly. However since SoundSieve is able to cover more informative blocks as mentioned in 6.2, the drop in classification performance due to change in energy harvesting pattern is minimal compared to other approaches.

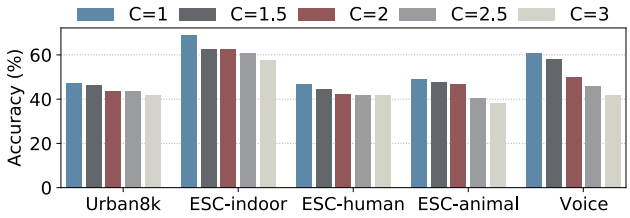
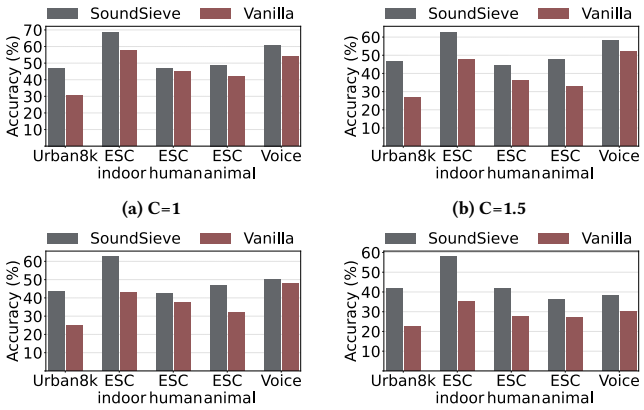


Figure 10: Effect of Harvesting Pattern

In Figure 11, we observe that SoundSieve achieves up to 25% higher classification accuracy than the uniform sampling strategy of the vanilla approach.

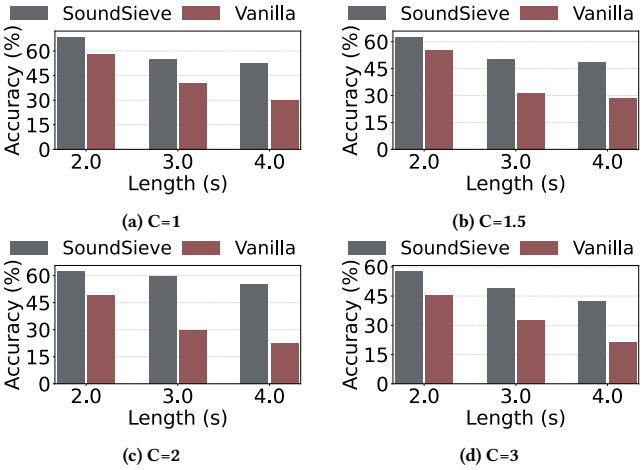


(a) C=1 (b) C=1.5

(c) C=2 (d) C=3

Figure 11: Comparison of SoundSieve and Vanilla approach at different energy harvesting rate

6.4 Effect of Audio Length

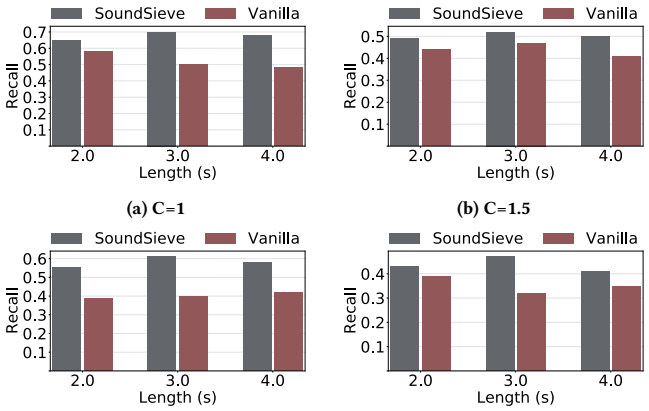


(a) C=1 (b) C=1.5

(c) C=2 (d) C=3

Figure 12: Effect of audio length at different energy harvesting rates.

We vary the length of audio clips from 2–4 seconds to test the robustness of SoundSieve. In Figure 12, we observe that SoundSieve outperforms vanilla across all energy harvesting rates for all audio



(a) C=1 (b) C=1.5

(c) C=2 (d) C=3

Figure 13: Comparison of recall of the most informative audio segments for different audio lengths and energy harvesting rates

lengths. This is because as the length of the audio increases, the intermittent system goes through more charge and discharge cycles, and consequently misses more audio segments. Sensing more informative segments becomes crucial as the length of the audio increases. This is evident in Figure 13 which compares their recall, i.e., the coverage of the most informative blocks by SoundSieve and the vanilla. The recall for SoundSieve remains consistent for audio clips of all lengths while it gets worse for the vanilla.

Note that 80% of the audio clips used in this experiment are about 2 seconds long and there are very few audio clips that are more than 3 seconds long. This creates an imbalance in the dataset and causes a drop in the accuracy as the audio length increases.

6.5 Effect of Segment Size

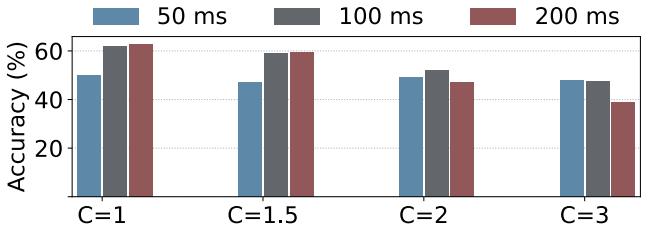


Figure 14: Performance of SoundSieve at different segment size

SoundSieve analyzes and processes audio in the units of fixed-length segments. In this experiment, we study how the segment size affects SoundSieve's classification accuracy under different energy harvesting conditions. We use the ESC-50 [61] dataset for this experiment. The result is summarized in Figure 14.

We observe that when the percentage of missing data is less, i.e., $C \leq 1.5$, the classification accuracy increases as we increase the length of each segment. This is because, with larger segments, the system gets a longer window of audio signals to compute the Fourier transform and extract useful acoustic features. In contrast, when the percentage of missing data is very high, e.g., $C > 2$, the system is highly likely to completely miss important segments if the segment length is large. Large segments give the system

less control over its sampling decisions, which causes a drop in classification accuracy when a large portion of the data is missing. This is why we see that the trend is reversed for higher values of C in Figure 14. From this experiment, we learn that it is better to use shorter segments when the percentage of missing data is very high, and larger segments when the percentage of missing data is low in an intermittent system.

6.6 Beyond Audio

Although SoundSieve is specifically designed for audio signals, we conduct a dataset-driven, limited-scale study to understand its potential in other types of signals.

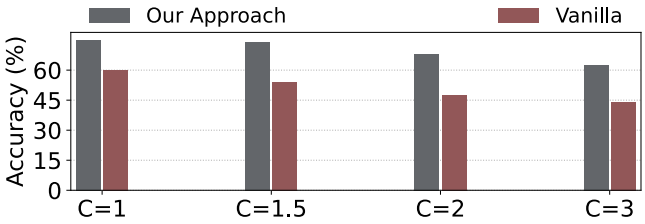


Figure 15: Comparison between our approach and the vanilla algorithm on UCI-HAR [4] dataset.

We use the UCI-HAR [4] dataset, which contains accelerometer and gyroscope readings from 30 participants performing six types of activities. The dataset is split into train (70%) and test (30%) sets. Figure 15 compares the classification accuracy of our approach with that of the vanilla approach for different values of C. We observe that our approach achieves 12% – 20% more accuracy than the Vanilla approach, and the gap increases when there are more missing data – showing the resilience of the proposed technique in extreme energy harvesting conditions.

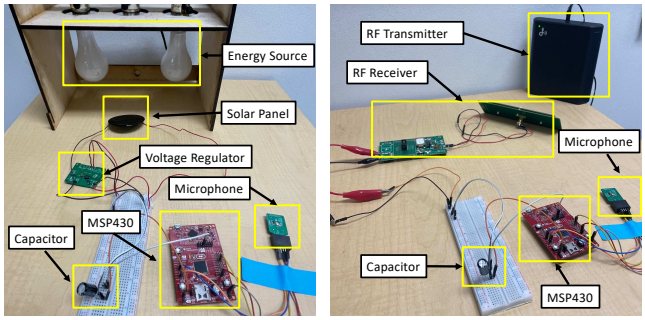
7 SYSTEM IMPLEMENTATION

7.1 Hardware Setup

We use TI MSP430FR5994 [2] MCU having 256KB FRAM, 8KB SRAM, direct memory access (DMA), and an operating voltage range of 1.8V to 3.3V at 8MHz CPU clock speed.

We use both solar and RF harvesters. The solar panel has polycrystaline solar cells [25] that outputs up to 5V at 40mA, which is regulated to 3.3V using [1]. The RF harvester consists of a TX91501 powercaster transmitter and a P2110 powerharvester receiver. A 470μF capacitor is used for energy storage.

A PMM-3738-VM1010-R piezoelectric MEMS microphone is used for audio sensing. This microphone has a zero-power listening mode which allows it to continuously sense for audio event triggers at extremely low-power. Whenever an audio event is triggered, the microphone sends an interrupt to the MSP430 and the microphone is set to active listening mode. During this active mode, the system continuously samples audio at 5KHz, performs FFT, executes the first convolution layer of the classifier in the frequency axis and writes the data to the FRAM using the low energy accelerator (LEA) and direct memory access (DMA).



(a) Solar Harvester Setup **(b) RF Harvester Setup**

Figure 16: Hardware Setup.

7.2 Embedded System Software

We modify an open source Python tool [76] to convert a TensorFlow model to a C header file which contains the weights and the neural network architecture. This header file is combined with platform-specific C implementation of neural network modules to obtain the audio classifier. We develop a complete C program that implements audio sensing, pre-processing, sampling, imputation, and classifier modules, and cross-compile to produce executable binary for the target system. We employ the static checkpoint calls similar to [3], where we profile the energy consumption at different states of sensing and when the capacitor voltage drops below 2.2V, the complete system state is copied to the FRAM.

7.3 Microbenchmarking

Execution Tracing. The intermittent system senses audio using a low-power microphone while it harvests energy from a RF harvester. We trace the operating voltage of the capacitor using a monitor [16]. After sensing each segment, SoundSieve decides whether to sense the next segment or to skip some segments, and wait in low-power mode to conserve energy and to fill up the energy buffer. This decision is made based on the current energy level, global segment importance, and local segment importance – predicted based on the current segment that is being observed.

A breakdown of all the steps of SoundSieve for classifying one audio clip is visualized in Figure 17. A breakdown of the sampling decisions is shown in Table 4. Initially, when there is no audio event to sense, the microphone remains in the low-power mode, and the system periodically wakes up and checks its energy harvesting rate. Using this information, a sampling strategy based solely on the global importance scores and energy harvesting rate is created. This step corresponds to the first row of Table 4. When an audio event triggers, the system wakes up and starts sensing and processing the first audio segment. At the end of processing this segment, the system has the local importance scores of next 5 blocks as well as the global sampling strategy. If the local importance score of a segment is higher than a threshold and there is enough energy to spend, the system samples that segment; otherwise, it waits for a more informative segment according to the global sampling strategy. For example, at time t_2, even though the x_5 has the highest importance score, the system decides to sense x_3 since its importance score was greater than the pre-determined threshold. After the audio event

ends, the system starts executing the DNN to classify the audio event.

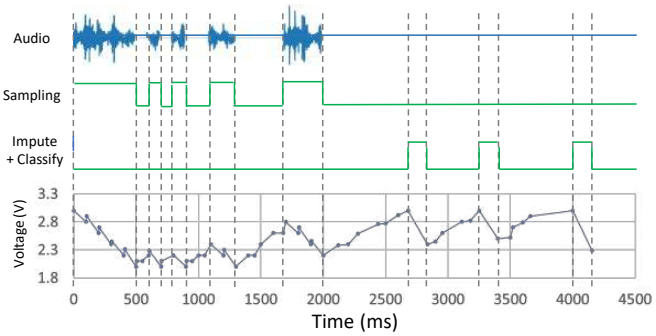


Figure 17: Execution steps of SoundSieve

Time	State	Segments by Importance	Decision
t_0	init	$(x_2, x_7, x_5, x_8, x_1, \cdots)$	Create Initial Sampling Strategy
t_1	sample	$(x_2, x_5, x_3, x_6, x_4)$	Sense at t_2
t_2	sample	$\{x_5, x_3, x_7, x_4, x_6\}$	Sense at t_3
t_3	sample	$\{x_7, x_4, x_8, x_5, x_6\}$	Sense at t_4
t_4	sample	$\{x_5, x_7, x_8, x_9, x_6\}$	Sense at t_5
t_5	sample	$\{x_8, x_7, x_9, x_{10}, x_6\}$	Wait until t_7
t_6	wait		Wait until t_7
t_7	sample	$\{x_9, x_8, x_{11}, x_{12}, x_{10}\}$	Wait until t_9
t_8	wait		Wait until t_9
t_9	sample	$\{x_{13}, x_{12}, x_{11}, x_{10}, x_{14}\}$	Wait until t_{12}
t_{10}	wait		Wait until t_{12}
t_{11}	wait		Wait until t_{12}
t_{12}	sample	$\{x_{13}, x_{16}, x_{14}, x_{17}, x_{15}\}$	Wait until t_{13}
t_{13}	sample	$\{x_{14}, x_{18}, x_{15}, x_{16}, x_{17}\}$	Wait until t_{18}

Table 4: Sampling strategy for an example audio clip

Overhead Measurement. Table 5 summarizes the time and energy overhead breakdown of on-device audio processing. Sensing and sampling decisions are made sporadically throughout the duration of the audio. The time overhead of the sampling algorithm is negligible since FFT and feature extraction for sampling decision are computed in parallel. The DNN runs when the microphone signals the end of audio. It takes less than a second (uninterrupted) to classify the audio—which can be interrupted 2-3 times due to intermittence.

Module	Time (ms)	Energy (μJ)
Sensing + Preprocess	100	510
Sampling Decision	7	60
DNN CONV (3 layers)	707	5850
DNN FC layer	5	100

Table 5: Execution time and energy overhead breakdown

8 EVALUATION ON REAL SYSTEM

8.1 Experimental Setup

Applications. After validating the sampling technique, imputation method, and the classifier using dataset-driven experiments, we design two real-world deployment scenarios using both solar and radio frequency (RF) harvesters. We implement two realistic applications: household activity detection, and voice phrase recognition. Each application has 10 classes of audio events and their lengths are shown in table 6. We use the DNN architecture shown in the Table 2 for classification. The DNN is split into three sub-networks to enable intermittent execution. After executing each sub-network, the intermediate results are stored in the FRAM. Then the system waits in the low-power mode while the capacitor charges, and periodically monitors the energy level. Finally, when enough energy is accumulated, the next sub-network of the DNN begins execution.

Application	Classes	Length (s)
Voice Phrases	10	1-3
Household Activities	10	2-4

Table 6: Two applications used for real system deployment. Voice phrases include ten different phrases such as "go left", "go right" etc. Household activities include sound events such as brushing teeth, pouring water, washing machine etc.

Methodology. For each application, we randomly select 30 clips for test – 1 clip for each of the 10 sound categories. We create a playlist with these 30 clips and play each clip from a mobile device every 10 second. Hence, the duration of the experiment is 300 seconds. Playing the sounds from a mobile device makes it possible to repeat the experiments and compare the results under different operation conditions such as mobility and energy variation.

To account for mobility and variation in harvested energy, we divide the 300 seconds into three 100 seconds phases. In the first phase, we keep both the sound source and the energy source fixed relative to the microphone. In the second and the third phases, we move the sound source and the energy source, respectively, at constant velocity. We use the solar setup for the voice application and the RF setup for the household application.

8.2 Experimental Results

Effect of Harvested Energy. In order to observe the effect of harvested energy we increase the distance between the RF harvester and the RF source when harvesting energy from RF, and change the light intensity when harvesting energy from solar. In Figure 18 between 100 seconds to 200 seconds, we observe that the classifier is able to maintain similar classification performance across both application for varying energy harvesting rates. During this period, we observe 4 out of 10 incorrect classifications for voice phrases and 3 out of 10 incorrect classifications for household activities, resulting in an accuracy of 60% and 67%, respectively.

Effect of Sound Source Mobility. We vary the distance between the sound source and the low-power microphone from 40 cm to 60 cm. The microphone is able to consistently wake up to the sound event and the classification accuracy remains similar as shown in Figure 18 from 200 seconds to 300 seconds. During this period, we observe 3 out of 10 incorrect classifications for voice phrases and 2 out of 10 incorrect classifications for household activities, resulting in an accuracy of 67% and 80%, respectively.

Effect of Audio Length. We experiment with audio clips of varying lengths – ranging from 1 second to 3 seconds for voice phrases, and 2 seconds to 4 seconds for household activity sounds. Since

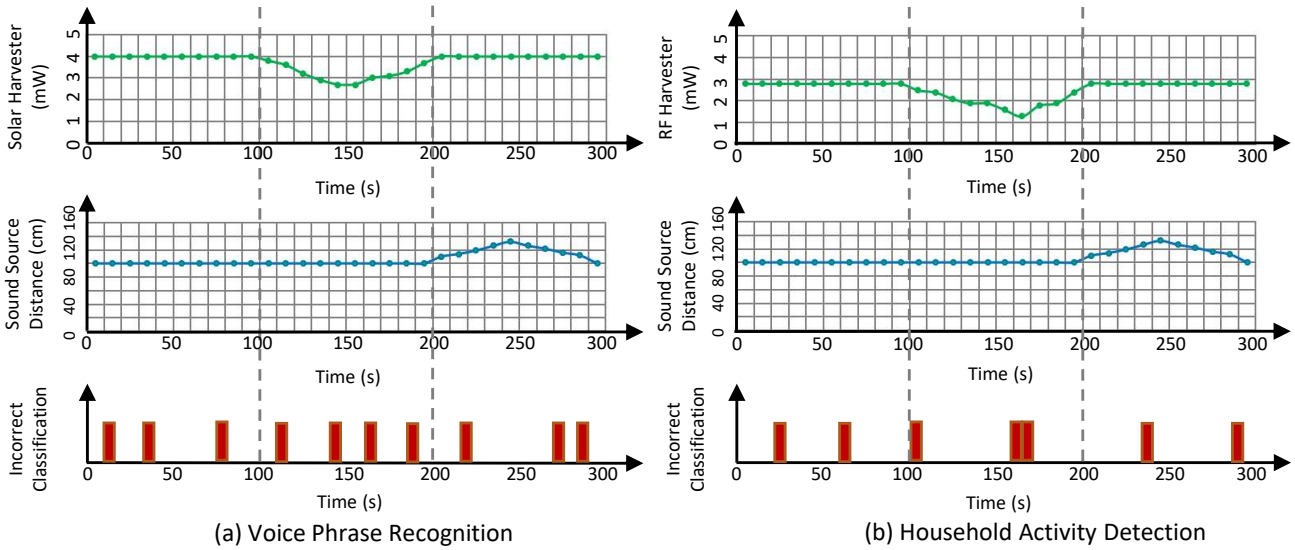


Figure 18: Effect of mobility and energy harvester in a real intermittent system.

SoundSieve is able to identify more informative segments in an audio, the classification performance remains consistently similar for audio clips of all lengths. Overall, we observe 10 out of 30 incorrect classifications for voice phrases and 7 out of 30 incorrect classifications for household activities, resulting in an accuracy of 67% and 76%, respectively.

9 RELATED WORK

Intermittent Computing. Intermittently powered systems experience frequent power failure that resets the software execution and results in repeated execution of the same code, and inconsistency in non-volatile memory. Previous works address the progress and memory consistency using software check-pointing [9, 24, 30, 43, 51, 54, 63, 69, 69], hardware interruption [5, 6, 55], atomic task-based model [13, 14, 50], non-volatile processors (NVP) [48, 49], and adaptive inference [36–38, 74].

Recently SONIC [19, 20, 34] proposes a unique software system for intermittent execution of deep neural inference combining atomic task-based model with loop continuation. None of the previous works perform intelligent sampling like SoundSieve.

Modeling Energy Harvesting Systems. [65] analytically model the trade-off associated with backing up data to maximize forward propagation. Even though energy harvesting system for a specific energy source has been analyzed and modeled before [15, 31, 66], none of the prior works focus on modeling the intermittence of sensor data on an intermittent system.

Intermittent Audio Sensing. In [52] authors demonstrated audio sensing in energy harvested devices, however, they only sense audio events of 283ms in length, which is not adequate for many real-world applications. In [39], authors simulated intermittency in audio sensing, however, they run post-processing and speech recognition networks on high-end devices without considering any computation and energy constraints.

Adaptive Informative Sampling. Resource constraint systems often need to monitor and sense large volume of data including video [8, 57], RF [18, 59] and audio [32]. In order to perform time and cost effective monitoring of the environment, many tiny robots and automated machines utilize adaptive informative sampling [18, 32, 33]. These robots model the environmental phenomena using Gaussian process regression (GPR) [72]. However GPR is an useful technique for modeling spatial uncertainty, not temporal uncertainty.

10 CONCLUSION

In this paper, we develop an intermittent audio sensing system that classifies audio events that are longer than the discharging cycle of the system. We leverage inherent properties of audio signals to design a lightweight audio sampler, an imputation method, and a classifier network to classify audio events. We show the effectiveness of the system using both dataset-driven and real-world evaluations. In the future, we plan to explore the possibility of designing content-aware intermittent sensing and classification systems that deal with complex data such as camera images.

ACKNOWLEDGMENTS

This work was supported, in part, by grants NIH 1R01LM013329-01 and NSF 2047461.

REFERENCES

[1] 2015. LTC3105 step up DC/DC converter. https://tinyurl.com/y6tp9ywf
[2] 2021. Msp430FR5994. http://www.ti.com/lit/ds/symlink/msp430fr5994.pdf
[3] Mikhail Afanasov, Naveed Anwar Bhatti, Dennis Campagna, Giacomo Caslini, Fabio Massimo Centonze, Koustabh Dolui, Andrea Maioli, Erica Barone, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, et al. 2020. Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 368–381.
[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.
[5] Domenico Balsamo, Alex S Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. 2016.

Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 12 (2016), 1968–1980.

[6] Domenico Balsamo, Alex S Weddell, Geoff V Merrett, Bashir M Al-Hashimi, Davide Brunelli, and Luca Benini. 2015. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters* 7, 1 (2015), 15–18.

[7] Fulvio Bambusi, Francesco Cerizzi, Yamin Lee, and Luca Mottola. 2022. The case for approximate intermittent computing. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 463–476.

[8] Jawadul H Bappy, Sujoy Paul, Ertem Tuncel, and Amit K Roy-Chowdhury. 2019. Exploiting typicality for selecting informative and anomalous samples in videos. *IEEE Transactions on Image Processing* 28, 10 (2019), 5214–5226.

[9] Naveed Bhatti and Luca Mottola. 2016. Efficient state retention for transiently-powered embedded sensing. In *International Conference on Embedded Wireless Systems and Networks*. 137–148.

[10] Naveed Anwar Bhatti and Luca Mottola. 2017. HarvOS: Efficient code instrumentation for transiently-powered embedded sensing. In *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on*. IEEE, 209–220.

[11] Bradford Campbell and Prabal Dutta. 2014. An energy-harvesting sensor architecture and toolkit for building monitoring and event detection. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*. 100–109.

[12] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. 2016. An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems. *ACM SIGOPS Operating Systems Review* 50, 2 (2016), 577–589.

[13] Alexei Colin and Brandon Lucia. 2016. Chain: tasks and channels for reliable intermittent programs. *ACM SIGPLAN Notices* (2016).

[14] Alexei Colin and Brandon Lucia. 2018. cc. In *Proceedings of the 27th International Conference on Compiler Construction*. ACM, 116–127.

[15] Andrea Crovetto, Fei Wang, and Ole Hansen. 2014. Modeling and optimization of an electrostatic energy harvesting device. *journal of microelectromechanical systems* 23, 5 (2014), 1141–1155.

[16] Digilent. 2022. Analog Discovery 2: 100MS/s USB Oscilloscope, Logic Analyzer and Variable Power Supply. https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/

[17] Guimin Dong, Mingyue Tang, Lihua Cai, Laura E Barnes, and Mehdi Boukhechba. 2021. Semi-supervised graph instance transformer for mental health inference. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1221–1228.

[18] Nicholas Fung, John Rogers, Carlos Nieto, Henrik I Christensen, Stephanie Kemna, and Gaurav Sukhatme. 2019. Coordinating multi-robot systems through environment partitioning for adaptive informative sampling. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 3231–3237.

[19] Graham Gobieski, Nathan Beckmann, and Brandon Lucia. 2018. Intermittent Deep Neural Network Inference. *SysML* (2018).

[20] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 199–213.

[21] Zdenek Hadas, Ludek Janak, and Jan Smilek. 2018. Virtual prototypes of energy harvesting systems for industrial applications. *Mechanical Systems and Signal Processing* 110 (2018), 152–164.

[22] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–13.

[23] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 17.

[24] Matthew Hicks. 2017. Clank: Architectural support for intermittent computation. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 228–240.

[25] Adafruit Industries. 2023. Round Solar Panel. https://www.adafruit.com/product/700

[26] Bahsima Islam, Yubo Luo, Seulki Lee, and Shahriar Nirjon. 2019. On-device training from sensor data on batteryless platforms. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. 325–326.

[27] Bashima Islam, Yubo Luo, and Shahriar Nirjon. 2023. Amalgamated Intermittent Computing Systems. *arXiv preprint arXiv:2303.13000* (2023).

[28] Bashima Islam and Shahriar Nirjon. 2019. Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems. *arXiv preprint arXiv:1905.03854* (2019).

[29] Bashima Islam and Shahriar Nirjon. 2020. Scheduling computational and energy harvesting tasks in deadline-aware intermittent systems. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 95–109.

[30] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. 2014. QuickRecall: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers. In *VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on*. IEEE, 330–335.

[31] Jinda Jia, Xiaobiao Shan, Deepesh Upadrashta, Tao Xie, Yaowen Yang, and Rujun Song. 2018. Modeling and Analysis of Upright Piezoelectric Energy Harvester under Aerodynamic Vortex-induced Vibration. *Micromachines* 9, 12 (2018), 667.

[32] Stephanie Kemna, David A Caron, and Gaurav S Sukhatme. 2016. Adaptive informative sampling with autonomous underwater vehicles: Acoustic versus surface communications. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 1–8.

[33] Stephanie Kemna, Hörur Heiarsson, and Gaurav S Sukhatme. 2018. On-board adaptive informative sampling for auvs: a feasibility study. In *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 1–10.

[34] Seulki Lee, Bashima Islam, Yubo Luo, and Shahriar Nirjon. 2019. Intermittent learning: On-device machine learning on intermittently powered system. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–30.

[35] Seulki Lee and Shahriar Nirjon. 2019. Neuro. zero: a zero-energy neural network accelerator for embedded sensing and inference systems. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 138–152.

[36] Yuyang Li, Yuxin Gao, Minghe Shao, Joseph T Tonecha, Yawen Wu, Jingtong Hu, and Inhee Lee. 2021. Implementation of multi-exit neural-network inferences for an image-based sensing system with energy harvesting. *Journal of Low Power Electronics and Applications* 11, 3 (2021), 34.

[37] Yuyang Li, Yawen Wu, Xincheng Zhang, Ehab Hamed, Jingtong Hu, and Inhee Lee. 2021. Developing a miniature energy-harvesting-powered edge device with multi-exit neural network. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[38] Yuyang Li, Yawen Wu, Xincheng Zhang, Jingtong Hu, and Inhee Lee. 2022. Energy-aware adaptive multi-exit neural network inference implementation for a millimeter-scale sensing system. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30, 7 (2022), 849–859.

[39] Yu-Chen Lin, Tsun-An Hsieh, Kuo-Hsuan Hung, Cheng Yu, Harinath Garudadri, Yu Tsao, and Tei-Wei Kuo. 2022. Speech Recovery For Real-World Self-Powered Intermittent Devices. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 26–30. https://doi.org/10.1109/ICASSP43922.2022.9747219

[40] Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2021. NeuroPose: 3D hand pose tracking using EMG wearables. In *Proceedings of the Web Conference 2021*. 1471–1482.

[41] Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2021. When video meets inertial sensors: Zero-shot domain adaptation for finger motion analytics with inertial sensors. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 182–194.

[42] Yilin Liu, Shijia Zhang, Mahanth Gowda, and Srihari Nelakuditi. 2022. Leveraging the Properties of mmWave Signals for 3D Finger Motion Tracking for Interactive IoT Applications. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 3 (2022), 1–28.

[43] Brandon Lucia and Benjamin Ransford. 2015. A simpler, safer programming and execution model for intermittent systems. *ACM SIGPLAN Notices* 50, 6 (2015), 575–585.

[44] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. 2019. Speech model pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:1904.03670* (2019).

[45] Yubo Luo and Shahriar Nirjon. 2019. Spoton: Just-in-time active event detection on energy autonomous sensing systems. *Brief Presentations Proceedings (RTAS 2019)* 9 (2019).

[46] Yubo Luo and Shahriar Nirjon. 2021. SmartON: Just-in-time active event detection on energy harvesting systems. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 35–44.

[47] Yubo Luo, Le Zhang, Zhenyu Wang, and Shahriar Nirjon. 2023. Efficient Multitask Learning on Resource-Constrained Systems. *arXiv preprint arXiv:2302.13155* (2023).

[48] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2017. Incidental computing on IoT nonvolatile processors. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 204–218.

[49] Kaisheng Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 526–537.

[50] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (2017), 96.

[51] Kiwan Maeng and Brandon Lucia. 2018. Adaptive Dynamic Checkpointing for Safe Efficient Intermittent Computing. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*.

[52] Amjad Yousef Majid, Patrick Schilder, and Koen Langendoen. 2020. Continuous Sensing on Intermittent Power. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 181–192. https://doi.org/10.1109/IPSN48710.2020.00-36

[53] Amjad Yousef Majid, Patrick Schilder, and Koen Langendoen. 2020. Continuous Sensing on Intermittent Power. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 181–192. https://doi.org/10.1109/IPSN48710.2020.00-36

[54] Azalia Mirhoseini, Ebrahim M Songhori, and Farinaz Koushanfar. 2013. Automated checkpointing for enabling intensive applications on energy harvesting devices. In *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*. IEEE, 27–32.

[55] Azalia Mirhoseini, Ebrahim M Songhori, and Farinaz Koushanfar. 2013. Idetic: A high-level synthesis approach for enabling long computations on transiently-powered ASICs. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 216–224.

[56] Cyan Subhra Mishra, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2021. Origin: Enabling on-device intelligence for human activity recognition using energy harvesting wireless sensor networks. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1414–1419.

[57] Mahathir Monjur, Iram Tazim Hoque, Tanzima Hashem, Md. Abdur Rakib, Judy E. Kim, and Sheikh Iqbal Ahamed. 2021. Smartphone based fundus camera for the diagnosis of retinal diseases. *Smart Health* 19 (2021), 100177. https://doi.org/10.1016/j.smhl.2020.100177

[58] Mahathir Monjur and Shahriar Nirjon. 2022. An Empirical Analysis of Perforated Audio Classification *(IASA '22)*. Association for Computing Machinery, New York, NY, USA, 25–30. https://doi.org/10.1145/3539490.3539602

[59] Sirajum Munir, Hongkai Chen, Shiwei Fang, Mahathir Monjur, Shan Lin, and Shahriar Nirjon. 2022. CarFi: Rider Localization Using Wi-Fi CSI. arXiv:2301.01592 [cs.NI]

[60] João Pedro Norenberg, João Victor Peterson, Vinicius Gonçalves Lopes, Roberto Luo, Leonardo de la Roca, Marcelo Pereira, José Geraldo Telles Ribeiro, and Americo Cunha Jr. 2021. STONEHENGE—Suite for nonlinear analysis of energy harvesting systems. *Software Impacts* 10 (2021), 100161.

[61] Karol J Piczak. 2015. ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*. 1015–1018.

[62] Archontis Politis, Sharath Adavanne, Daniel Krause, Antoine Deleforge, Prerak Srivastava, and Tuomas Virtanen. 2021. A Dataset of Dynamic Reverberant Sound Scenes with Directional Interferers for Sound Event Localization and Detection. In *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*. Barcelona, Spain, 125–129. https://dcase.community/workshop2021/proceedings

[63] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2012. Mementos: System support for long-running computation on RFID-scale devices. *Acm Sigplan Notices* 47, 4 (2012), 159–170.

[64] J. Salamon, C. Jacoby, and J. P. Bello. 2014. A Dataset and Taxonomy for Urban Sound Research. In *22nd ACM International Conference on Multimedia (ACM-MM'14)*. Orlando, FL, USA, 1041–1044.

[65] Joshua San Miguel, Karthik Ganesan, Mario Badr, and Natalie Enright Jerger. 2018. The EH model: Analytical exploration of energy-harvesting architectures. *IEEE Computer Architecture Letters* 17, 1 (2018), 76–79.

[66] Himanshu Sharma, Ahteshamul Haque, and Zainul Jaffery. 2018. Modeling and Optimisation of a Solar Energy Harvesting System for Wireless Sensor Network Nodes. *Journal of Sensor and Actuator Networks* 7, 3 (2018), 40.

[67] Lukas Sigrist, Andres Gomez, Matthias Leubin, Jan Beutel, and Lothar Thiele. 2020. Environment and application testbed for low-power energy harvesting system design. *IEEE Transactions on Industrial Electronics* 68, 11 (2020), 11146–11156.

[68] Yue Tang, Yawen Wu, Peipei Zhou, and Jingtong Hu. 2022. Enabling Weakly Supervised Temporal Action Localization From On-Device Learning of the Video Stream. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 3910–3921.

[69] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 17–32.

[70] Zhepeng Wang, Yawen Wu, Zhenge Jia, Yiyu Shi, and Jingtong Hu. 2021. Lightweight run-time working memory compression for deployment of deep neural networks on resource-constrained MCUs. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 607–614.

[71] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).

[72] Andrew Gordon Wilson, David A Knowles, and Zoubin Ghahramani. 2011. Gaussian process regression networks. *arXiv preprint arXiv:1110.4411* (2011).

[73] Yawen Wu and Jingtong Hu. 2022. Towards Independent On-device Artificial Intelligence. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 351–351.

[74] Yawen Wu, Zhepeng Wang, Zhenge Jia, Yiyu Shi, and Jingtong Hu. 2020. Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[75] Yawen Wu, Zhepeng Wang, Yiyu Shi, and Jingtong Hu. 2020. Enabling on-device cnn training by self-supervised instance filtering and error map pruning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3445–3457.

[76] Le Zhang, Yubo Luo, and Shahriar Nirjon. 2022. Demo Abstract: Capuchin: A Neural Network Model Generator for 16-bit Microcontrollers. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 497–498.

[77] Shijia Zhang, Yilin Liu, and Mahanth Gowda. 2022. Let's Grab a Drink: Teacher-Student Learning for Fluid Intake Monitoring using Smart Earphones. In *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE Computer Society, 55–66.

[78] Shijia Zhang, Yilin Liu, and Mahanth Gowda. 2023. I Spy You: Eavesdropping Continuous Speech on Smartphones via Motion Sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–31.

[79] Hao Zhou, Taiting Lu, Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2022. Learning on the Rings: Self-Supervised 3D Finger Motion Tracking Using Wearable Sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–31.