Batteryless Gesture Recognition Via Learned Sampling

Geffen Cooper
The University of Texas at Austin
geffen@utexas.edu

Radu Marculescu
The University of Texas at Austin
radum@utexas.edu

***Abstract*—Batteryless wearables offer the potential for continuous and maintenance-free operation through ambient energy harvesting. However, the constrained and uncertain nature of harvested energy makes it impossible to *continuously* sense, process, and transmit data. In this batteryless setting, we no longer have access to complete and continuous data streams. Thus, we must adapt machine learning models to this new paradigm where only a limited *subset* of the data can be sampled and processed. In this work, we consider solar-powered gesture recognition as a target application where sensing and processing are constrained to a strict energy budget. We propose a learning-based approach where a sampling policy and gesture classifier are jointly trained via a shared representation. This enables the system to learn which sensor samples are informative while simultaneously optimizing the classifier for sparse inputs. By actively deciding when to sample, we improve gesture recognition accuracy by up to 10% compared to fixed-rate subsampling across a wide range of energy budgets. This closes the gap to a standard battery-powered approach by up to 40%.**

***Index Terms*—Wearables, batteryless sensors, deep learning**

I. Introduction and Prior Work

Batteryless wearables powered by ambient energy harvesting can enable a new class of sustainable sensing systems [1]. By eliminating batteries, these devices have become appealing for long-term health monitoring [2], activity recognition [3], and human-computer interaction [4]. To explore wearable batteryless systems from an *algorithmic* perspective, we target gesture recognition as a representative task. Gesture recognition enables natural interaction between users and devices with applications ranging from assistive technologies to virtual reality, making it a critical task for wearable computing [5].

While gesture recognition with battery-powered devices such as smartwatches is well-established [5], removing the stability of a continuous power source fundamentally changes the problem. Specifically, in batteryless systems, energy is highly constrained, making it impossible to guarantee the sampling of complete gesture sequences. This leads to sparse, and energy-dependent data, breaking the assumptions behind conventional models trained on full gesture sequences.

Existing batteryless gesture recognition works tend to focus on novel hardware for low-power sensing [6]–[9], but rely on standard data processing pipelines with fixed length inputs. In contrast, we use an off-the-shelf accelerometer and adopt a learning-based approach that explicitly addresses the challenge of energy constrained sampling by learning *when* to sample and how to classify the resulting sparse input sequences.

Beyond gesture recognition, prior work on machine learning with batteryless sensors has explored early exit networks [10] and intermittent computing [11] to enable battery-free, on-device inference. However, these works assume fully observed inputs. In contrast, we optimize which portions of the input signal to sample. Others have investigated energy-aware data acquisition strategies [3], where a policy decides when to spend or conserve energy to improve over opportunistic baselines. Our approach differs in two key ways: (1) we jointly optimize the sampling policy and classifier within a unified model, and (2) we operate under a finite time-horizon regime, where the system strategically allocates a fixed energy budget over the uncertain duration of a gesture. **Overall, our contributions are as follows**:

1) We design a learning framework that jointly optimizes sampling and classification via a shared representation.
2) We evaluate our approach across a range of sampling budgets on two gesture datasets, achieving up to 10% higher accuracy over baseline strategies.
3) We validate the feasibility and practicality of our problem setting by profiling a physical hardware prototype.

The rest of the paper is organized as follows: Section II defines our problem setting and approach. Section III outlines our experiments, results, and hardware validation. Finally, Section IV outlines our main contributions. We publish our code at: `https://github.com/SLDGroup/LearnedSampling`

II. Methodology

A. Problem Setting

Battery-powered Setting: Gesture recognition operates on a sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ of samples, where $\mathbf{x}_k \in \mathbb{R}^d$ is a sample at time step k and the length n is variable. In our work, we use a 3-axis accelerometer ($d = 3$) where a sample $\mathbf{x}_k$ has (X, Y, Z) components. Each gesture $\mathbf{X}$ has a label $Y \in \{1, \dots, C\}$ where C is the number of gesture classes. The goal is to learn a classifier $\hat{Y} = f_\theta(\mathbf{X})$ parametrized by θ from a dataset $(\mathbf{X}, Y)$, where $\hat{Y}$ is the model prediction.

In our setting, we assume the *onset* of a gesture ($k = 1$) is identified as the time when the acceleration magnitude surpasses a given threshold. In a traditional battery-powered system, the full gesture would then be densely sampled and processed by a classifier f. However, in a *batteryless* setting, dense sampling is not feasible, prompting the need for selective sampling and processing strategies.

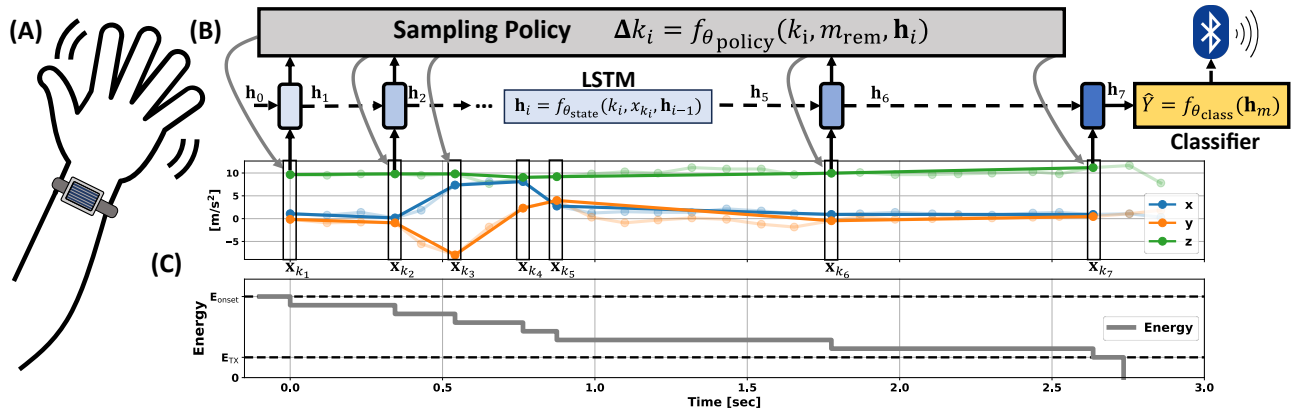

Fig. 1. **(A) Overview.** A wearable on the wrist harvests solar energy to power an accelerometer. At the onset of a gesture, our system has stored energy E_{onset} which corresponds to a fixed number of observations m for gesture recognition. **(B) Sampling and Processing.** With each observed sample $\mathbf{x}_{k_i}$, we use an LSTM $f_{\theta_{\text{state}}}$ to accumulate information into a learned state $\mathbf{h}_i$ where i is the observation index and k_i is the time step. At each decision point, the hidden state $\mathbf{h}_i$, current time k_i, and remaining budget m_{rem} are used by the policy $f_{\theta_{\text{policy}}}$ (gray box) to determine Δk_i, the delay before observing the next sample (gray arrows). Between samples, the system sleeps and uses negligible energy. Once the budget m is exhausted, we pass the final hidden state $\mathbf{h}_m$ to the classifier $f_{\theta_{\text{class}}}$. The gesture prediction $\hat{Y}$ is then transmitted via Bluetooth (BLE). **(C) Sampled Signal.** In the top plot, the faded lines show the full accelerometer data, while the dark lines show the sampled (observed) data. In the bottom plot, we see that a fixed amount of energy E_{update} is used to observe and process each new sample. The dashed lines show the budget for sampling and processing with the remaining budget being used for transmission (E_{TX}).

Batteryless Setting: Our system harvests solar energy for sensing, processing, and communication. In our setting, the instantaneous solar power harvested is sufficient to power the accelerometer for detecting the gesture onset, but it is **not** sufficient to sense and process samples for gesture recognition. Thus, energy is buffered into a capacitor until the gesture onset, at which point our system uses the available energy to selectively sample a *subset* of the data $\tilde{\mathbf{X}} = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \ldots, \mathbf{x}_{k_m})$ where $k_i \in \{1, \ldots, n\}$ is a non-uniform time step and m is the sampling budget. This procedure is outlined in Figure 1.

B. Energy Constrained Sampling

At the gesture onset, our system has a fixed energy budget, E_{onset}, and sensing becomes a sequential decision making process where each sampling decision is informed by the data collected so far. To this end, we use an LSTM (Long Short-Term Memory) [12], a recurrent neural network which maintains a *hidden state* to capture information over time. Overall, our system consists of three jointly learned parts $\boldsymbol{\theta}_{\text{system}} = \{\theta_{\text{state}}, \theta_{\text{policy}}, \theta_{\text{class}}\}$ where θ_{state}, θ_{policy}, and θ_{class} are the parameters of the LSTM ($f_{\theta_{\text{state}}}$), sampling policy ($f_{\theta_{\text{policy}}}$), and gesture classifier ($f_{\theta_{\text{class}}}$) respectively. With each observed sample $\mathbf{x}_{k_i}$, our LSTM updates the hidden state:

$$\mathbf{h}_i = f_{\theta_{\text{state}}}(k_i, \mathbf{x}_{k_i}, \mathbf{h}_{i-1}) \tag{1}$$

Note, we use k_i as an input to add temporal context since the samples are not uniformly spaced. The updated hidden state is then used by the policy to determine when to sample next:

$$\Delta k_i = f_{\theta_{\text{policy}}}(k_i, m_{\text{rem}}, \mathbf{h}_i) \tag{2}$$

where m_{rem} is the remaining sampling budget, and Δk_i is the number of samples to delay for until observing the next sample ($k_{i+1} = k_i + \Delta k_i$). The final hidden state $\mathbf{h}_m$ is then used by the classifier as $\hat{Y} = f_{\theta_{\text{class}}}(\mathbf{h}_m)$. The output Δk_i is predicted from the set $\{1, 2, 4, 8, 16\}$ which consists of powers of two to provide a compact yet flexible range of temporal resolutions. We define the number of samples the system can observe as the sampling budget m, which is a function of the initial energy, E_{onset}, the cost to read **and** process one accelerometer sample, E_{update}, and the cost to transmit the prediction over BLE, E_{TX}:

$$m = \left\lfloor \frac{E_{\text{onset}} - E_{\text{TX}}}{E_{\text{update}}} \right\rfloor \tag{3}$$

C. Jointly Optimizing Sampling and Classification

As shown in Figure 1, the hidden state of the LSTM represents the shared representation used by our sampling policy and gesture classifier. Ideally, the policy learns to map this evolving representation to sampling decisions that yield the most 'informative' and discriminative samples, enabling the classifier to accurately distinguish between gestures.

Pretraining with Random Sampling: Learning the LSTM representation is challenging as all modules have randomly initialized weights. Initially, the classifier cannot provide meaningful feedback, and the policy has no signal about which samples are useful. Thus, we pretrain the LSTM and classifier using a random policy. This provides a preliminary representation and classifier that allow the policy to learn which samples are most important for classification.

Policy Optimization as Supervised Learning: Even with a pretrained representation and classifier, directly learning the policy is challenging as we lack labels indicating when to sample. Thus, we cast the problem as a supervised learning task. The goal is to learn a function that maps the current hidden state $\mathbf{h}_i$ (at decision step i) to the delay Δk until the next sample $\mathbf{x}_{k_{i+1}}$ as shown in Equation (2). We now explain how we learn this function via Algorithm 1.

1. Initialization: At the start of Algorithm 1, we assume we have a pretrained LSTM representation and classifier which were trained using a random policy. Then, for each budget m, we train a randomly initialized policy for sampling.

2. Main Algorithm: For each decision index i within the budget, we aim to train the policy to map the hidden state $\mathbf{h}_i$ to the best subsequent observation time k_{i+1}. Thus, in the main part of the algorithm (Lines 2-6), we train the policy to predict the next best sampling time when starting from decision index i, and then finetune the LSTM and classifier representations.

3. Policy Training: To train the policy in a supervised manner (Lines 7-11), we need to find the next best sampling time from decision step i for each gesture in the dataset. We use this *best delay* as a label and optimize the parameters of the policy with the Adam optimizer [13] to predict this delay.

4. Supervision via Rollouts: To find the best delay (Lines 12-21), we roll out the policy to decision step i (Line 14). From k_i, we sample multiple candidate delays (1,2,4, ... steps), and then use *uniform subsampling* to consume the *remaining* budget. By keeping the *remainder* of each trajectory comparable, we isolate the impact of the single delayed observation. Intuitively, small delays lead to denser sampling early in the gesture, while larger delays preserve more budget for denser sampling later. Finally, we train the policy to predict the delay that led to the highest confidence of ground truth gesture Y.

5. Finetuning the Representation: After training the policy for the current delay index, we freeze it and finetune the LSTM representation and the classifier using standard supervised learning. The only difference is that the forward pass through the LSTM uses the frozen policy to sample the input.

Algorithm 1 Jointly Optimizing Sampling and Classification

Require: m, sampling budget; $(\theta_{\text{state}}^{\text{pretrain}}, \theta_{\text{class}}^{\text{pretrained}})$, pretrained LSTM and classifier; $\{(\mathbf{X}, Y)\}$, a gesture dataset;

1: initialize θ_{policy}
2: **for** $i = 1$ to m **do**
3: $\quad\boldsymbol{\theta}_{\text{system}} \leftarrow \{\theta_{\text{state}}, \theta_{\text{policy}}, \theta_{\text{class}}\}$
4: $\quad\theta_{\text{policy}} \leftarrow \textsc{TrainPolicy}(\boldsymbol{\theta}_{\text{system}}, m, i, \{(\mathbf{X}, Y)\})$
5: $\quad\theta_{\text{state}}, \theta_{\text{class}} \leftarrow \textsc{FineTune}(\boldsymbol{\theta}_{\text{system}}, m, i, \{(\mathbf{X}, Y)\})$
6: **end for**
7: **function** $\textsc{TrainPolicy}(\boldsymbol{\theta}_{\text{system}}, m, i, \{(\mathbf{X}, Y)\})$
8: $\quad Y_{\text{policy}} \leftarrow \textsc{BestDelay}(\boldsymbol{\theta}_{\text{system}}, m, i, (\mathbf{X}, Y))$
9: $\quad\theta_{\text{policy}} \leftarrow \text{AdamUpdate}(\theta_{\text{policy}}, \{(\mathbf{X}, Y_{\text{policy}})\})$
10: $\quad$**return** θ_{policy}
11: **end function**
12: **function** $\textsc{BestDelay}(\boldsymbol{\theta}_{\text{system}}, m, i, (\mathbf{X}, Y))$
13: $\quad\mathbf{h} \leftarrow \mathbf{0}$, $m_{\text{rem}} \leftarrow m$, $k \leftarrow 0$
14: $\quad\mathbf{h}_i, k, m_{\text{rem}} \leftarrow \text{LSTM\_Policy}(\mathbf{X}, i, \mathbf{h})$
15: $\quad$**for** $l \in \{1, 2, 4, 8, 16\}$ **do**
16: $\quad\quad\text{Traj}_l \leftarrow \left[\mathbf{X}[k+l, :], \mathbf{X}[k+l :: \Delta_{\text{unif}}, :]\right]$
17: $\quad\quad\text{pred}_l \leftarrow f_{\theta_{\text{class}}}(\text{LSTM}(\text{Traj}_l, \mathbf{h}))$
18: $\quad$**end for**
19: $\quad Y_{\text{policy}} \leftarrow \underset{l}{\text{argmax}}\{\text{pred}_l[Y]\}$
20: $\quad$**return** Y_{policy}
21: **end function**
22: **function** $\textsc{FineTune}(\theta_{\text{state}}, \theta_{\text{class}}, m, i, \{(\mathbf{X}, Y)\})$
23: $\quad\hat{Y} \leftarrow f_{\theta_{\text{class}}}$ (LSTM_Policy$(\mathbf{X}, i, \mathbf{h})$)
24: $\quad\theta_{\text{state}}, \theta_{\text{class}} \leftarrow \text{AdamUpdate}(\theta_{\text{state}}, \theta_{\text{class}}, \{(\mathbf{X}, Y)\})$
25: $\quad$**return** θ_{state}, θ_{class}
26: **end function**

III. Experiments

A. Experimental Details

Datasets: We use the **SmartWatch Gesture** [14] (20 gestures, 8 subjects) and **Motion Gesture** [15] (6 gestures, 14 subjects) datasets which were collected with a smartwatch. Since we introduce a new problem formulation, we evaluate against three baselines. *(1) Random Sampling*: select m random time steps. *(2) Uniform Subsampling*: select m time steps by uniformly subsampling. Since the gesture length varies, we subsample based on the average length. *(3) Dense Sampling*: use all samples with no energy constraint (the upper bound).

Training: We use the default PyTorch LSTM (hidden size 32). The policy and classifier are fully connected layers that take the hidden state as input. We train the *Dense*, *Random*, and *Uniform Subsampling* models for 300 epochs with the Adam optimizer (batch size 8, learning rate 0.0005), and use dropout (rate 0.5) for the classifier. For finetuning and policy training (Algorithm 1), we use a learning rate of 0.01, batch size of 8, and 5 epochs per iteration (Line 2 of Algorithm 1).

Evaluation: We use the macro-average F1-score as our evaluation metric and use leave-One-Subject-Out-Cross-Validation (LOSOCV), training on $S - 1$ subjects and testing on the remaining one where S is the number of subjects. We repeat the experiment across 3 seeds and test our approach across a range of sampling budgets $m \in \{2, 3, 4, 5, 6, 7\}$.

B. Results

Figures 2 and 3 show the mean F1-scores (averaged over seeds and subjects) per budget, with error bars for standard deviation. The dashed line shows the dense sampling upper bound. On the **SmartWatch** dataset, *Uniform Subsampling* (orange) outperforms *Random Sampling* (blue) as random sampling can miss key gesture segments whereas uniform subsampling weights all segments equally. However, on the **Motion** dataset, which contains high variability in gesture length, random sampling performs similarly or better. This is because uniform subsampling uses a fixed sampling rate based on the average sequence length, causing it to underutilize the budget on short gestures or run out of samples before the gesture ends on long gestures. In contrast, our learned policy (green) adapts to the input, consistently achieving the highest F1-score and approaching the upper bound with larger budgets.

C. Hardware Validation

We profile a hardware prototype (Fig. 4) consisting of an nRF52832 (MCU) and BMA400 accelerometer. A solar panel charges a 100 uF capacitor via a DFM8001 power management chip (PMIC) which delivers 1.8V to the PCB. The BMA400's internal logic wakes the MCU when motion is detected.

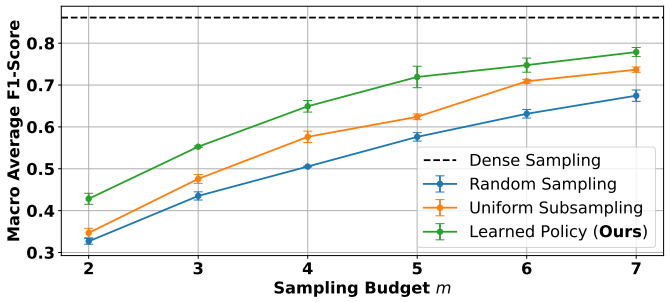

Fig. 2. **SmartWatch Gesture** dataset results. We plot the mean with standard deviation error bars of the Macro-F1-Score across 6 sampling budgets. The upper dashed line shows the mean F1 for dense sampling.

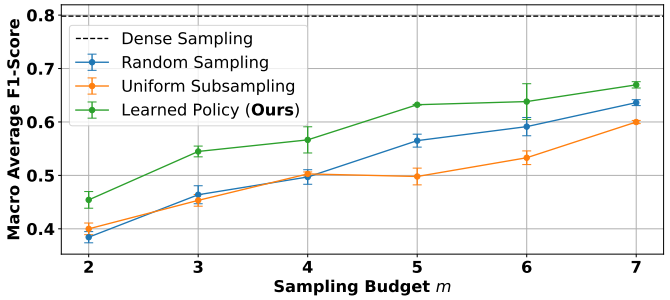

Fig. 3. **Motion Gesture** dataset results. We plot the mean with standard deviation error bars of the Macro-F1-Score across 6 sampling budgets. The upper dashed line shows the mean F1 for dense sampling.

Indoor light is enough to power the motion detection logic: [(2.5uA MCU sleep) + (0.85uA BMA400 logic)] $\cdot$ 1.8V $\approx 6uW$. The DFM8001 charges the capacitor until 2.7V and turns off once it discharges to 2.2V giving $\frac{1}{2}$100uF$(2.7^2 - 2.2^2) \approx$ 120uJ of usable energy. Profiling shows an average cost of $E_{\text{TX}} = 25$uJ (1 byte payload) and $E_{\text{update}} = 15$uJ. Applying Equation (3), $m = \lfloor (120 - 25)/15 \rfloor = 6$ samples in the best case which validates our problem setting. Importantly, our method adds negligible cost. Breaking down E_{update}: reading one sample (at 25 Hz) costs 3uJ, the LSTM state update (Eq. (1)) costs 12uJ, and the policy evaluation (Eq. (2)) adds $\approx$ 0.2uJ. Since every policy requires an LSTM update for classification, our method only adds the negligible cost of policy evaluation, keeping Equation (3) valid across policies.

IV. Conclusion

In this paper, we developed a learning-based framework for gesture recognition in the batteryless setting, where only a limited subset of the input can be observed. By jointly training a sampling policy and gesture classifier via a shared representation, our approach learns to select informative samples with negligible overhead. Our method achieves the best performance across two datasets and 6 sampling budgets when averaged across multiple subjects and random seeds. Finally, we validated our problem setting using a physical prototype, showing that batteryless gesture recognition is feasible with commodity hardware. In future work we seek to generalize our approach to use a single model for all sampling budgets.

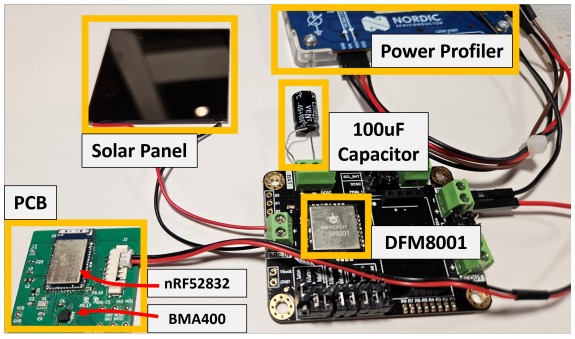

Fig. 4. **Hardware Prototype**. A solar panel charges a 100uF capacitor via a power management chip which delivers a regulated 1.8V to the PCB.

Acknowledgment

This work was supported in part by NSF Grant ECCS-2428656 and the iMAGiNE Consortium at UT Austin. We also thank David Sisson for assistance with the PCB design.

References

[1] S. Ahmed *et al.*, "The internet of batteryless things," *Communications of the ACM*, vol. 67, no. 3, 2024.

[2] H. Kim, B. Rigo, G. Wong, Y. J. Lee, and W.-H. Yeo, "Advances in wireless, batteryless, implantable electronics for real-time, continuous physiological monitoring," *Nano-Micro Letters*, vol. 16, no. 1, 2024.

[3] G. Cooper and R. Marculescu, "Packet pruning: Finding better energy spending policies for batteryless human activity recognition," in *2024 IEEE 20th International Conference on Body Sensor Networks (BSN)*. IEEE, 2024, pp. 1–4.

[4] A. Alsubhi *et al.*, "User-centered perspectives on the design of battery-less wearables," *International Journal of Human–Computer Interaction*, vol. 40, no. 23, pp. 8025–8046, 2024.

[5] R. Tchantchane, H. Zhou, S. Zhang, and G. Alici, "A review of hand gesture recognition systems based on noninvasive wearable sensors," *Advanced intelligent systems*, vol. 5, no. 10, p. 2300207, 2023.

[6] H. Truong *et al.*, "Capband: Battery-free successive capacitance sensing wristband for hand gesture recognition," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 54–67.

[7] A. Jayatilaka and D. C. Ranasinghe, "Real-time fluid intake gesture recognition based on batteryless uhf rfid technology," *Pervasive and Mobile Computing*, vol. 34, pp. 146–156, 2017.

[8] M. Kodali, S. Sigg *et al.*, "Towards battery-less rf sensing," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2021, pp. 352–355.

[9] Y. Li, T. Li, R. A. Patel, X.-D. Yang, and X. Zhou, "Self-powered gesture recognition with ambient light," in *Proceedings of the 31st annual ACM symposium on user interface software and technology*, 2018.

[10] P. Farina *et al.*, "Memory-efficient energy-adaptive inference of pre-trained models on batteryless embedded systems," *arXiv preprint arXiv:2405.10426*, 2024.

[11] L. Caronti, K. Akhunov, M. Nardello, K. S. Yıldırım, and D. Brunelli, "Fine-grained hardware acceleration for efficient batteryless intermittent inference on the edge," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5, pp. 1–19, 2023.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] L. Porzi, S. Messelodi, C. M. Modena, and E. Ricci, "A smart watch-based gesture recognition system for assisting people with visual impairments," in *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices*, 2013, pp. 19–24.

[15] R.-D. Vatavu and O.-C. Ungurean, "Understanding gesture input articulation with upper-body wearables for users with upper-body motor impairments," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–16.