From Raw Waveforms to On-Device Earthquake Detection: Real-Time Seismic Data Analysis for MCUs

TAYYABA ZAINAB, Distributed Systems, Kiel University, Kiel, Germany and Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, Germany
PATRICK RATHJE, Distributed Systems, Kiel University, Kiel, Germany
LAURA HARMS, Distributed Systems, Kiel University, Kiel, Germany and Institute for Networked Cyber Physical Systems, Hamburg University of Technology, Hamburg, Germany
LUKAS SCHATTENHOFER, Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, Germany and Distributed Systems, Kiel University, Kiel, Germany
JENS KARSTENS, Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, Germany
OLAF LANDSIEDEL, Institute for Networked Cyber Physical Systems, Hamburg University of Technology, Hamburg, Germany and Distributed Systems, Kiel University, Kiel, Germany

Detecting earthquakes in seismological time series is a core task in observational seismology, supporting a range of applications from early warning systems to tectonic research. Typically, seismic sensors passively record data and send it to the cloud or edge for integration, storage, and analysis. While this cloud-based approach is effective in urban or well-connected areas, it is impractical in remote, underwater, or underground environments where network infrastructure is unreliable. In such settings, the sensors must operate independently for extended periods while coping with strict constraints on power, memory, and connectivity. To address these challenges, we present LightEQ, a system that combines an efficient data processing pipeline and a lightweight deep-learning model specifically designed for seismic event detection in such environments. LightEQ runs on ultra-low-power microcontrollers with just 100 kB of RAM, enabling real-time, on-device earthquake detection without the need for continuous streaming of raw data to a central location. We evaluate LightEQ against a traditional STA/LTA approach and state-of-the-art (SOTA) machine learning models, using the Stanford Earthquake Dataset. Unlike existing neural network (NN) models, which are too large for microcontrollers, LightEQ is over ten times smaller than most of the SOTA models. Our results demonstrate that communication is the most energy-intensive task in this setting, and that traditional model-driven filters like STA/LTA are inefficient due to their high false positive rate. In contrast, LightEQ improves detection

This work was funded in part by the Helmholtz School for Marine Data Science (MarDATA) under grant No. HIDSS–0005 and by the project KI-gesteuertes Monitoring mariner Mikroalgen als CO_2-Senke (KIMMCO) under grant No. 3719481010.
Authors' Contact Information: Tayyaba Zainab (corresponding author), Distributed Systems, Kiel University, Kiel, SH, Germany and Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, SH, Germany; e-mail: tayyaba.zainab@cs.uni-kiel.de; Patrick Rathje, Distributed Systems, Kiel University, Kiel, SH, Germany; e-mail: pra@informatik.uni-kiel.de; Laura Harms, Distributed Systems, Kiel University, Kiel, Germany and Institute for Networked Cyber Physical Systems, Hamburg University of Technology, Hamburg, Germany; e-mail: laura.harms@cs.unikiel.de; Lukas Schattenhofer, Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, SH, Germany and Distributed Systems, Kiel University, Kiel, SH, Germany; e-mail: lschattenhofer@geomar.de; Jens Karstens, Marine Geodynamics, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, SH, Germany; e-mail: jkarstens@geomar.de; Olaf Landsiedel, Institute for Networked Cyber Physical Systems, Hamburg University of Technology, Hamburg, Germany and Distributed Systems, Kiel University, Kiel, Germany; e-mail: olaf.landsiedel@tuhh.de.

ACM 2577-6207/2026/04-ART32
https://doi.org/10.1145/3799716

accuracy with NN, providing a more energy-efficient solution by reducing the number of false positives before transmission. Compared to the STA/LTA method alone, LightEQ extends battery life by at least 3-fold by minimizing energy consumption associated with transmitting false positives to the cloud.

CCS Concepts: • **Computer systems organization → Neural networks**; *Real-time system architecture*; **Embedded software**; • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Seismological data analysis, earthquake detection, on-device, deep neural networks, low-power, internet of things, TinyML, Edge AI

ACM Reference Format:
Tayyaba Zainab, Patrick Rathje, Laura Harms, Lukas Schattenhofer, Jens Karstens, and Olaf Landsiedel. 2026. From Raw Waveforms to On-Device Earthquake Detection: Real-Time Seismic Data Analysis for MCUs. *ACM Trans. Internet Things* 7, 3, Article 32 (April 2026), 33 pages. https://doi.org/10.1145/3799716

1 Introduction

A dense global network of seismic sensors monitors the Earth's subsurface to better understand natural phenomena such as earthquakes, volcanic eruptions, and landslides, as well as anthropogenic subsurface activities such as oil and gas extraction, CO_2 storage sites, and hydrothermal energy production [16]. Most seismic sensors continuously transmit their raw data to a central facility, where the data is analyzed to detect and extract relevant seismic events for further analysis. This cloud-based approach is effective in urban or well-connected areas, but it is impractical in harsh environments such as underwater, subterranean, or remote locations with unreliable network infrastructure. In these settings, communication is the most energy-consuming task, far outweighing the cost of sensing or processing [13, 40]. As these sensors operate on limited energy reserves, continuously transmitting raw seismic data is not feasible if a long system lifetime is desired. Since seismic events are rare and the majority of recorded data consists of background noise, transmitting all data is highly inefficient. Instead, our goal is to identify seismic events on the device and only communicate relevant ones, enabling energy-efficient filtering and selective data transfer. This conserves power and extends the operational lifetime of seismic monitoring systems, without sacrificing data quality or requiring frequent physical access.

This article extends the design and evaluation of LightEQ [61], a lightweight deep learning pipeline to run on tiny, low-power devices such as microcontrollers (MCUs). LightEQ enables on-device analysis of recorded data from seismic sensors to detect seismic events such as earthquakes. By detecting seismic events from raw data on the MCU, LightEQ significantly reduces the amount of data that needs to be transmitted, compared to sending all raw sensor data. The pipeline begins with a pre-filter, namely, a Short-Time Average versus Long-Time Average (STA/LTA) filter [29], applied to the continuously sampled sensor data to efficiently identify potential earthquake events [30]. The filtered data is then converted into a time-frequency map using the Short-Time Fourier Transform (STFT) and processed by a lightweight NN (see Figure 1). STA/LTA, a simple pre-filter, is prone to generating a high number of false positives. LightEQ enhances detection accuracy by incorporating the NN to reduce these false positives generated during the STA/LTA pre-filtering step. Although pre-filtering adds processing overhead, it drastically reduces the frequency of NN execution and results in substantial energy savings. For example, combining a pre-filter with NN classification to suppress irrelevant data before transmission significantly extends sensor lifetime by at least 3-fold compared to transmitting all pre-filtered events using STA/LTA alone.

For the NN, we design a set of lightweight quantized models tailored for resource-constrained devices in the Internet of Things (IoT), enabling deployment of the complete pipeline on MCUs, such as the Cortex-M4, with as little as 100 kB of RAM. In this article, we evaluate the NN with the

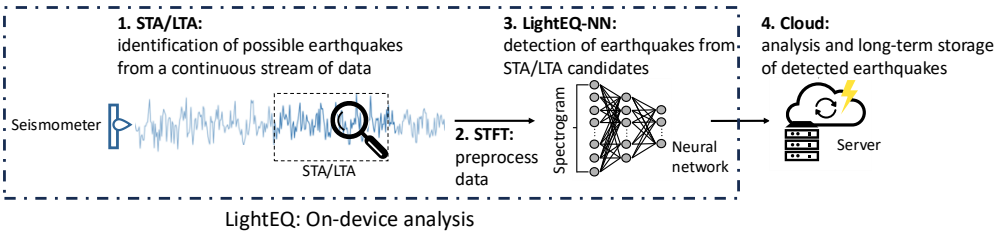


Fig. 1. Overview of LightEQ: A lightweight STA/LTA filter continuously monitors a raw seismic data stream and identifies potential earthquake events. When a frame is flagged, it is preprocessed using STFT and passed to a lightweight NN for on-device classification. After classification, the event and associated metadata can be transmitted to the cloud over a low-bandwidth connection for long-term storage or further analysis. Together, the STA/LTA filter and LightEQ-NN enable comprehensive on-device event detection.

latest software optimizations, resulting in faster inference times and reduced memory usage. These improvements further reinforce the suitability of LightEQ for deployment on resource-constrained MCUs. We achieve at least 13-fold improvement in inference time on Cortex-M4 by using the built-in digital signal processor (DSP) available on modern MCUs for hardware-accelerated inference compared to float inference, and we also improve the inference time by 2-fold compared to our previous work [61]. Our smallest model with 29k parameters achieves 99% accuracy. This NN, along with STFT, consumes only 1.9 µWh when processing one minute of raw data on a Cortex-M4. In this article, we use this model as part of our pipeline, which includes receiving periodic sensor measurements over a serial interface (e.g., UART), filtering obvious noise, transforming the data to a time-frequency map, and executing the NN on a resource-constrained MCU. However, receiving data at higher frequencies generates more interrupts and keeps the MCU in active mode, significantly increasing energy consumption. To mitigate this overhead, we further evaluate the use of larger UART buffers, which accumulate data at the same sampling frequency but process it at lower frequencies. In our baseline setup, processing occurs 100 times per second. Reducing the processing rate to once per second through buffering extends the sensor's operational lifetime by 73%.

We evaluate LightEQ against state-of-the-art (SOTA) models, including CRED [37], EQTransformer [34], LCANet [63], LEQNet [28], and SeismicSense [60], using the Stanford Earthquake Dataset (STEAD) [35]. Unlike existing neural network (NN) models, LightEQ is over ten times smaller than the smallest comparable model and remains deployable on MCU. SeismicSense [60] also offers competitive performance and is the only other SOTA NN model deployable on MCUs besides LightEQ. However, due to its more complex architecture, it requires five times more energy and has a five-fold longer execution time compared to LightEQ. We release LightEQ as open source to the public.[1]

Contributions. Our previous work [61] on LightEQ focuses primarily on the NN component. In this article, we extend the LightEQ's design, implementation, and evaluation to include additional pipeline components beyond the NN. Furthermore, we optimized and deployed the complete LightEQ pipeline on a Cortex-M4 MCU, evaluating its performance with pre-recorded seismic data on resource-constrained devices. We summarize the contributions of this work as follows:

(1) We introduce a data processing pipeline that starts with a pre-filter to identify potential earthquake events, which are then preprocessed and verified by a lightweight NN. This demonstrates the feasibility of on-device NN-based earthquake classification on resource-constrained IoT devices.

[1]Implementation available at: https://github.com/ds-kiel/LightEQ_pipeline_extended.git

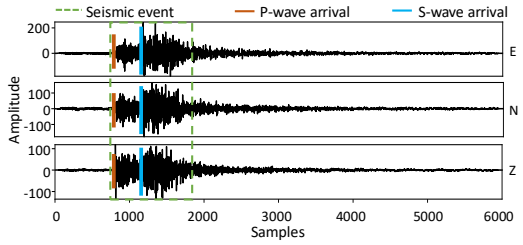


Fig. 2. Three-component seismic waveform (Z: vertical, N: north-south, E: east-west) recorded during an earthquake. The traces show the characteristic onset of a P-wave followed by an S-wave. These recordings reflect typical patterns used in seismic event analysis.

(2) We present a set of lightweight earthquake detection models for MCUs with as little as 100 kB of RAM. These models achieve 99% accuracy in detecting earthquakes in the STEAD dataset, with the smallest model having only 29k parameters.
(3) We extend the design and evaluation, compared to the previous conference version, to investigate the impact of combining pre-filtering with the NN. This approach extends battery life by at least 3-fold compared to the standalone pre-filter. The evaluation also includes exploring different STA/LTA configurations and analyzing tradeoffs in accuracy, computational cost, and memory usage.
(4) We evaluate our NNs on MCUs and report a minimum of 13-fold reduction in inference time on Cortex-M4 using DSP acceleration over float models, and a 2-fold improvement over previous work.
(5) We demonstrate the end-to-end deployment of the complete pipeline in a real-world setting, covering sensor data acquisition through to processing. By reducing the processing rate from every 10 milliseconds to once per second, we extend the operational lifetime of a Cortex-M4–based device by approximately 73%.

Outline. We provide background in Section 2 and related literature in Section 3. Section 4 details our processing pipeline: sensor data reception, pre-filtering, preprocessing, and NN classification, including lightweight model derivation and implementation details. Section 5 introduces datasets, and an in-depth evaluation of LightEQ and LightEQ-NN. Finally, we conclude in Section 6.

2 Background

This section outlines the key concepts and methods used in this article. We begin by explaining seismic data and its role in monitoring geological activity. We then describe the STA/LTA algorithm for seismic anomalies, followed by the STFT.

2.1 Seismic Data

Seismic waves, generated by subsurface processes like tectonic fault activity and magma movement, carry crucial information about Earth's internal structure and dynamics. These waves are essential for studying natural phenomena, such as volcanic eruptions, often preceded by numerous earthquakes due to magma intrusion and crustal pressure increases [9], as well as anthropogenic subsurface activities like oil and gas extraction or CO_2 geological storage [47]. Seismometers record these ground motions, typically capturing three components: vertical (Z), north/south (N), and east/west (E) movements [26]. Earthquakes primarily generate two types of body waves: faster, compressional P-waves, which travel through solids and fluids, and slower, shear S-waves, which propagate only through solids and follow P-waves [47]. The arrival of P-waves, followed by S-waves, together constitute the seismic signal of an event, as illustrated in Figure 2. Seismic sensors record

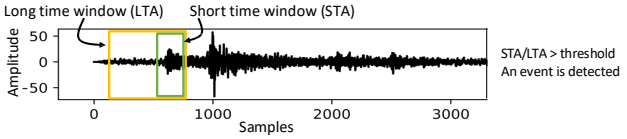


Fig. 3. When the resulting ratio exceeds a predefined threshold, the corresponding data segment is flagged as a potential earthquake event and forwarded to the preprocessing block, followed by NN-based classification for verification.

these ground motions as time-series data, providing continuous temporal traces for each component (Z, N, E). This data allows for identifying signal patterns, including seismic phase arrival times, amplitude variations, and event durations, which are fundamental for subsequent tasks like event detection and classification [34].

2.2 Short-Term Average/Long-Term Average (STA/LTA)

The STA/LTA algorithm [30] is a standard technique in seismological monitoring for detecting seismic events. It compares the average energy of a short leading window (STA) with a long trailing window (LTA) for each seismic channel (see Figure 3). STA identifies sudden signal changes, indicating potential earthquakes, while LTA tracks background noise. The ratio is computed by squaring amplitudes to represent energy, averaging over both windows, and taking their quotient [1]. This is formalized in the equation below:

$$\text{STA/LTA}(t) = \frac{\frac{1}{N_s} \sum_{i=t-N_s+1}^{t} x^2(i)}{\frac{1}{N_l} \sum_{i=t-N_s-N_l+1}^{t-N_s} x^2(i)}.$$

Here, $x(i)$ is the seismic amplitude at time i, N_s is the short-term window size, and N_l is the long-term window size. A high STA/LTA ratio at time t suggests a potential event onset. If this ratio exceeds a user-defined threshold, an alert is triggered. Selecting this threshold is crucial: setting it too low results in frequent false alarms, while setting it too high may cause true events to be missed. In our pipeline, we use STA/LTA as a pre-filter to detect potential earthquakes in the data stream [29, 49]. However, the traditional STA/LTA algorithm's O(n) computational cost per sample for recomputing full window averages is inefficient. Instead, we use a recursive STA/LTA formulation [45, 46], which incrementally updates the short-term average S_t and long-term average L_t. This reduces complexity to O(1) per sample, significantly improving efficiency for real-time detection on low-power embedded devices. The recursive updates are:

$$S_t = S_{t-1} + \frac{1}{N_s}\left(x^2(t) - x^2(t-N_s)\right)$$

$$L_t = L_{t-1} + \frac{1}{N_l}\left(x^2(t-N_s) - x^2(t-N_s-N_l)\right)$$

$$\text{STA/LTA}(t) = \frac{S_t}{L_t}.$$

2.3 Frequency Spectrogram Using Short-Time Fourier Transform

The Fourier Transform (FT) computes the amplitudes of frequencies within a given signal. As earthquakes usually have characteristic frequencies, different from background noise [5], analyzing

the frequencies' amplitudes helps to characterize earthquake signals [48]. The STFT computes a time-localized frequency map, i.e., the frequency amplitudes for individual, shorter windows: STFT segments the signal into smaller time intervals, computes a FT for each segment, and integrates the amplitude spectra for each time step into a 2D frequency versus time matrix referred to as a spectrogram or frequency-time map [18]. While the traditional FT only identifies the occurrence of an event, the STFT provides additional information by pinpointing the exact start time of the signal, allowing for a more detailed analysis of its temporal characteristics (see Figure 16).

3 Related Work

In the following, we discuss the SOTA of traditional model-driven approaches and the younger field of data-driven approaches for earthquake detection.

3.1 Model-Driven Analysis

One basic technique for event detection in seismic signals is "threshold triggering", which detects events based on a user-defined threshold. However, this type of technique is highly sensitive to noise and does not work in many environments [46, 58]. For example, in rural environments, transient ground vibrations caused by livestock movement, agricultural machinery, or wind-blown vegetation can frequently trigger false positives, making threshold-based detection impractical. A more robust approach is the STA/LTA method [53, 58], discussed in Section 2. STA/LTA performs well with impulsive, high Signal-to-Noise Ratio (SNR) signals. However, it is inefficient under conditions of low SNR or increased ambient noise, which can lead to false positives [45]. Like Liu at el [29], we also use STA/LTA as a pre-filter for anomaly detection. It causes many false positives on noisy data, but also virtually zero false negatives if configured sufficiently sensitive. Another approach is "template matching" [58], also known as filter matching, which uses waveform similarity to perform a sensitive comparison of a candidate waveform with a template waveform. An event is detected if the normalized correlation coefficient is above a certain threshold. More sophisticated algorithms, such as the fingerprinting and similarity threshold algorithm (FAST) [58], use locality sensitive hashing to improve the efficiency of the similarity search. Lately, these traditional methods have been replaced by data-driven approaches and, most recently, by NNs, as described in the following sections.

3.2 Data-Driven Analysis

In recent years, machine learning has become an integral part of seismological data analysis [8, 34, 37]. For example, Chin et al. (2019) [8] utilize three machine learning algorithms for earthquake detection: K-nearest neighbor, classification tree, and support vector machine. K-nearest neighbor is the simplest earthquake classifier and calculates the Euclidean distance between the new observation and the feature record. A majority vote of Top K shortest distance records drives this classification. The classification tree is built based on the training data, where each branch of the tree is a feature cutoff point that guides the decision to a potential class. Finally, their support vector machine determines the hyperplane that classifies the training data into two groups with the maximum margin in the high-dimensional feature space. While these are lightweight approaches in terms of computational complexity and memory demands, they do not achieve the accuracy of modern approaches based on NNs.

3.2.1 Neural Networks for the Cloud. Recent research proposes using deep convolutional neural network [41, 43, 52, 57], or combined convolutional and recursive neural network (CNN-RNN) architectures like CRED [37], to capture both spatial and temporal features from seismic spectrograms. CRED, for instance, utilizes a deep residual CNN with 14 convolutional layers and two bi-directional long short-term memory (LSTM) layers, achieving a 98% F_1 score. Other CNN-based models, such as PhaseNet [64] and PickNet [56], focus on P- and S-wave picking but can also be

Table 1. Summary of State-of-the-art Models in Contrast to LightEQ, including their Tasks, Memory Requirements, and Deployability on MCUs. The Upper Portion of the Table Lists Event-level Detection Methods that are Directly Comparable to LightEQ. The Lower Portion of the Table Presents Models that Address the More Complex Task of Phase Detection and are Therefore not Directly Comparable to LightEQ, but are included for Contextual Completeness

Models	Model size	Task		DNN Detection	MCU
		Event Detection	Phase Detection		
STA/LTA [30]	-	✓		-	✓
CRED [37]	3.7 MB	✓		RNN	
LightEQ	0.09 MB	✓		RNN	✓
Phasenet [64]	3.2 MB		✓	CNN	
Picknet [56]	72 MB		✓	CNN	
EQTransformer [34]	5.2 MB	✓	✓	Attention	
LCANet [63]	3.7 MB	✓	✓	Attention	
LEQNet [28]	0.94 MB	✓	✓	Attention	
SeismicSense [60]	0.18 MB	✓	✓	RNN	✓

adapted for general event detection. EQTransformer [34] employs attention-based deep learning [54] for earthquake and P- and S-wave identification in cloud environments, achieving close to 100% accuracy. However, these cloud-based approaches are resource-intensive and unsuitable for deployment on resource-constrained edge devices, as we show in our evaluation in Section 5.

3.2.2 Neural Networks for the Edge. Edge devices bridge the gap between cloud servers and MCUs, offering a platform for lightweight NN models. Unlike the cloud's abundant resources, edge deployments necessitate architectures that minimize memory and computational demands while maintaining accuracy. LCANet [63], inspired by SqueezeNet [19], reduces parameters by 50% using 1x1 convolutional filters while achieving performance comparable to AlexNet [24]. LCANet specifically optimizes deep learning models for edge devices using context-aware attention modules. Other edge-focused projects include NetQuakes [33], which detects earthquakes using MEMS sensors, and Seo et al.'s work [45] using autoencoder-based deep learning with MEMS sensors on Korean high-speed trains. LEQNet [28] proposes a deep NN that reduces model size by 87.68% compared to EQTransformer [34] and achieves an F_1 score of 99%, but its transformer architecture makes it impractical for MCU deployment due to high resource demands [14]. Despite significant progress in designing efficient models for edge devices, these architectures remain too large or computationally intensive for deployment on MCUs, which have much stricter memory and power constraints compared to edge devices. This highlights the need for even more compact and optimized models specifically tailored to ultra-low-power environments.

3.2.3 Neural Networks for the Sensor Plane. To bridge the gap between edge-level models and the constraints of MCU-based deployment, recent efforts have shifted toward designing models that can run directly at the sensor level. This challenge is not unique to seismology. Similar constraints arise in acoustic event detection [6], ship or vessel detection [15, 27], structural health monitoring [12, 59], and keyword spotting [21]. These applications demand that on-device NNs operate within strict resource budgets. In seismology, other than LightEQ, SeismicSense [60] is the only other notable MCU-deployable NN. Like LightEQ, it employs an STA/LTA pre-filter to reduce NN execution, but it further identifies P- and S-wave arrivals through a modular design. Table 1 summarizes this section, highlighting that only STA/LTA and SeismicSense are deployable on MCUs.

4 Design: LightEQ

Designing an earthquake detection pipeline for embedded devices requires addressing three core constraints: limited computational resources, real-time processing demands, and restricted communication bandwidth. These factors shape LightEQ's overall design:

(1) **Resource Limitations:** Embedded devices offer limited processing power, memory, and energy. Continuously running NN models is challenging due to high computational demands, while power-efficient model-driven approaches often compromise accuracy. In addition, operation in remote locations further emphasizes the need for power-efficient solutions to ensure long-term functionality.
(2) **Data Processing Under Latency Constraints:** Raw seismic data requires low-latency processing to avoid missing new samples. This demands a highly efficient pipeline that minimizes buffering delays and meets strict timing requirements for continuous data streams on low-power hardware.
(3) **Low Communication Bandwidth:** In infrastructure-limited environments (e.g., rural, underwater), low-bandwidth connections (e.g., a few kilobits per second [44]) make continuous raw data transmission impractical. Instead, devices must detect and verify seismic events locally, transmitting only critical data. This necessitates accurate on-device filtering to reject false positives while ensuring true events trigger communication.

To meet these constraints, LightEQ efficiently classifies seismic data on embedded devices, distinguishing earthquakes from noise with accuracy comparable to SOTA models despite stricter resource constraints. This section covers LightEQ's architectural overview (Section 4.1), the preprocessing pipeline (Section 4.2) up to the NN (Section 4.3.1), system design (Section 4.4), and implementation details (Section 4.5).

4.1 LightEQ Overview

Earthquake detection using NNs offers high accuracy, as discussed in Section 3. However, it remains computationally demanding, even with lightweight models. Since earthquakes are rare events and represent only a small fraction of the total seismic data, processing all incoming seismic data with NN is inefficient. Our evaluation shows that analyzing each minute of raw seismic data with the NN consumes significantly more energy compared to traditional signal processing methods. These findings imply that continuous NN inference is not sustainable on energy-constrained devices. To address this, LightEQ performs near real-time analysis of incoming seismic signals (within one sampling interval, i.e., 10 msec) by incorporating an efficient pre-filtering step that triggers the NN only when necessary. This section details the system design of LightEQ, a four-step on-device data analysis pipeline that processes raw seismic signals through to final classification (Figure 4). We start with a light pre-filtering using the STA/LTA algorithm on the continuously recorded data from the seismometer. This step eliminates obvious noise samples and ensures that only potential earthquake signals are passed to the NN component for further analysis. The NN processes a time-frequency map, generated by the STFT, from the pre-filtered seismic signals. The NN outputs a prediction vector corresponding to smaller time intervals, called subframes. These predictions are then passed through a threshold filter, which converts the NN outputs into a binary classification, identifying each subframe as either noise or an earthquake signal based on a predefined threshold.

4.2 LightEQ Preprocessing Pipeline

As outlined in Section 4.1, the LightEQ pipeline consists of four stages, from raw data acquisition to final classification. In this article, we use the term preprocessing pipeline to refer specifically to the stages up to the generation of the NN input, i.e., data sampling, STA/LTA filtering, and the STFT

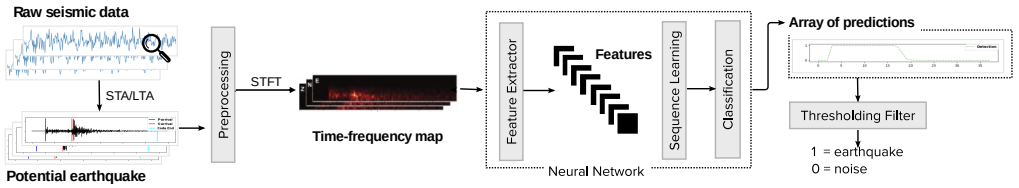


Fig. 4. System design of LightEQ: We continuously monitor raw seismic data using STA/LTA filters to detect potential earthquake signals. When the STA/LTA is triggered, we send this seismic data to a pre-processing unit, which converts it into a time-frequency map to support feature extraction. We then feed it into a NN, which processes it through convolutional and recurrent layers and outputs a prediction vector. Finally, a threshold filter processes this vector to produce a binary decision.

transformation. Below, we outline key design considerations for optimizing each preprocessing component.

(1) **STA/LTA Threshold Selection:** The STA/LTA algorithm serves as a preliminary filter for seismic activity. Tuning its threshold involves a tradeoff: a lower threshold increases sensitivity but risks frequent, unnecessary NN activations, while a higher one reduces false positives but might miss weak earthquake signals. We empirically optimize this to balance reliability and efficiency.
(2) **STA/LTA's Memory Consumption:** LTA calculation necessitates storing a sliding window of past seismic data, typically 10–30 seconds long, requiring 4 to 12 kB per channel at 100 Hz and 32-bit precision [1, 23]. To optimize memory on resource-constrained MCUs, we utilize the STA window within the LTA window, eliminating the need for separate memory allocation for the STA window.
(3) **STFT Computational Complexity:** The STFT, involving segmentation, windowing, and Fourier transforms, is computationally intensive. A typical multi-channel signal creates many overlapping windows, each needing an FFT and generating numerous frequency values. To reduce this load, we accelerate STFT operations using an on-chip DSP and optimize memory via in-place processing.

4.2.1 Preprocessing Steps. We now describe the design and implementation of each component in the preprocessing pipeline, beginning with the raw seismic signal acquisition, and proceeding through the STA/LTA trigger and STFT feature extraction.

Seismometer Readings. LightEQ receives continuous input from a SOTA digital 3-axis seismometer that records ground motion along the north-south (N), east-west (E), and vertical (Z) directions at a sampling rate of 100 Hz [23, 31, 55]. This high sampling frequency yields 300 measurements per second across all channels, providing the fine-grained temporal resolution necessary for detecting the rapid onset of seismic events. The raw data is streamed in real-time to the embedded device, where it enters the preprocessing pipeline. Processing this continuous input stream prevents the embedded device from entering low-power sleep modes, which significantly increases power consumption even when no earthquakes occur. To address this, we use duty-cycling for the processing of seismometer readings instead of processing them continuously, thus, reducing power consumption. While reducing the processing rate from once every 10 milliseconds to once per second introduces latency, selecting a higher processing frequency helps maintain near real-time responsiveness. In our evaluation in Section 5.6, we evaluate processing intervals of 1, 10, 100, and 1,000 samples to show the tradeoffs between latency, memory, and energy efficiency.

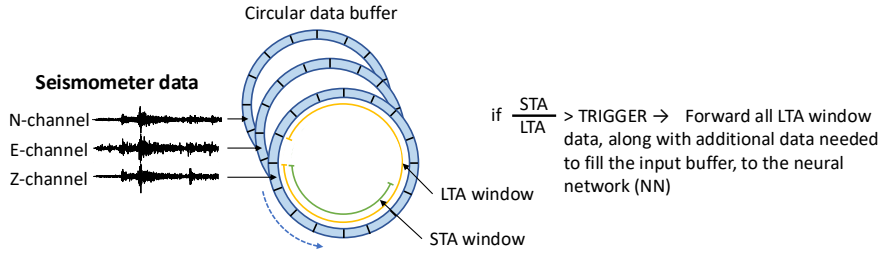


Fig. 5. We begin our pipeline with STA/LTA analysis on all incoming data across N, E, and Z channels. If any channel surpasses the trigger threshold, the system forwards the relevant LTA window along with extra data to ensure the input buffer is ready for our NN to process. This approach ensures thorough, multi-channel monitoring before activating the deep learning stage.

Table 2. To Optimize LightEQ's Preprocessing Pipeline for Detection Sensitivity and Reduce False Triggers, We Perform a Grid Search Over Key STA/LTA Parameters

Parameter	Range	Increment	Purpose
STA Window (samples)	100–2,000	100	To experiment with sensitivity to short-term signal changes.
LTA Window (samples)	1,000–4,000	250	To account for a wide range of background noise levels.
Trigger Threshold (STA/LTA Ratio)	1.2–6.0	0.2	To fine-tune detection sensitivity across varying noise environments and event magnitudes.

STA/LTA. We apply STA/LTA as lightweight filters to each of the three input channels (see Figure 5). We use three circular buffers (one per channel), each sized to the LTA window and storing the most recent samples. To manage the limited memory of the embedded device, we opt for a design where the STA window lies within the LTA window and corresponds to its most recent samples, thereby avoiding separate memory allocation for STA. Furthermore, we use recursive computations for both STA and LTA to minimize computational overhead (see Section 2) to support near real-time responsiveness. To balance sensitivity and false positive rate, we systematically experiment with a range of STA/LTA trigger ratios and window lengths. Specifically, we constrain STA to be at most half the length of LTA [53] (see Table 2) and set the STA and LTA ranges based on the observed variability of P-wave arrival times in our dataset, which typically fall between 500 and 1000 samples after the start of the event. This configuration ensures that the STA window captures the initial energy buildup while the LTA provides a stable long-term reference. We experimented with different STA/LTA settings using threshold values between 1.2 and 6.0, allowing us to balance high recall with effective suppression of noise-induced triggers. The trigger ratio strongly influences the algorithm's responsiveness to seismic events. Our evaluation (see Section 5.3) analyzes how this parameter impacts detection performance, highlighting the tradeoff between sensitivity to true events and the rate of false positives.

STFT. After STA/LTA triggers, we apply STFT to convert the relevant segment of the seismic time series into a time-frequency representation. To construct the STFT input, we extract the 750 most recent samples per channel from the existing LTA circular buffer and collect additional samples per channel in real-time, forming a 6,000-sample per channel window, equivalent to one minute of data at a 100 Hz sampling rate. We include the 750 values from the LTA window to ensure the trigger is part of the input. This design captures both the data leading up to the trigger and the subsequent

seismic activity. Figure 16 illustrates the STFT output for a single seismic channel from two different events. In each subfigure, the top plot shows the raw time-domain signal, while the lower plot displays the corresponding time-frequency representation. The x-axis denotes time, the y-axis frequency, and color intensity reflects the STFT magnitude. STFT outputs three independently computed time-frequency maps for the three channels and then stacks along the channel dimension to form a 3-channel time-frequency representation. This structured input is passed to LightEQ-NN for seismic event detection, as detailed in the following section.

4.3 LightEQ-NN

After the preprocessing pipeline has identified a potential earthquake and transformed the data into the time-frequency space, this section focuses on the last but key part of our pipeline: the NN component of LightEQ for earthquake detection. We discuss the design considerations associated with our NN architecture that delivers high detection accuracy while operating within the strict resource constraints of embedded devices, followed by the introduction of its base architecture, which serves as the foundation for subsequent architecture search and hyperparameter optimization to obtain a set of lightweight models. Designing a NN for MCU-based platforms requires careful balancing of accuracy, model complexity, and inference latency. Below, we highlight the key factors that influence the architecture and deployment of our NN models:

(1) **Balancing Model Complexity with Limited Resources:** Designing NNs for resource-constrained devices requires optimization of layer configurations. Increasing the number of parameters can improve accuracy, but it often comes at the cost of higher memory and computational demands. For example, CNN layers with strides greater than one reduce parameters but risk losing subtle features, especially for short earthquake events (detailed in Section 5.8).
(2) **Managing Inference Time and Memory Constraints:** Minimizing inference time is also crucial for real-time earthquake monitoring. Larger models typically increase latency. While RNNs, especially LSTMs, effectively model seismic data and improve accuracy (see Section 3), they also add many parameters, which increase model size and inference time. Flash memory usage scales directly with the number of parameters, whereas RAM requirements depend on intermediate activations during inference and are therefore less predictable, often needing to be measured directly on the device.
(3) **Quantization and Deployment Tradeoffs:** Smaller models often meet memory and latency constraints, but at the cost of accuracy. Rather than limiting our design space to such compact models, we first develop architectures with strong baseline performance. Although these models are initially too large for MCU deployment, post-training quantization reduces their size by up to 4-fold, with a typical accuracy loss of 2–20% due to reduced precision and error accumulation in RNN layers [22]. Even with this degradation, the quantized models achieve higher accuracy than very small unquantized models.

These design tradeoffs inform the structure of our NN architecture. Next, we present the baseline architecture that is the foundation for our subsequent architecture and hyperparameter search.

4.3.1 Baseline Neural Network Architecture. The baseline architecture of LightEQ-NN is inspired by CRED [37], but we significantly extend and adapt the framework to meet the strict resource constraints of embedded devices (see Figure 6). LightEQ-NN is trained as a regression model, where each output element represents the estimated probability of earthquake activity within a specific sub-interval of the input waveform (see Figure 11). A post-processing thresholding function, independent of the NN, counts the number of predicted activations and produces a final binary decision of earthquake or no earthquake. We use a dual-threshold mechanism from CRED with an

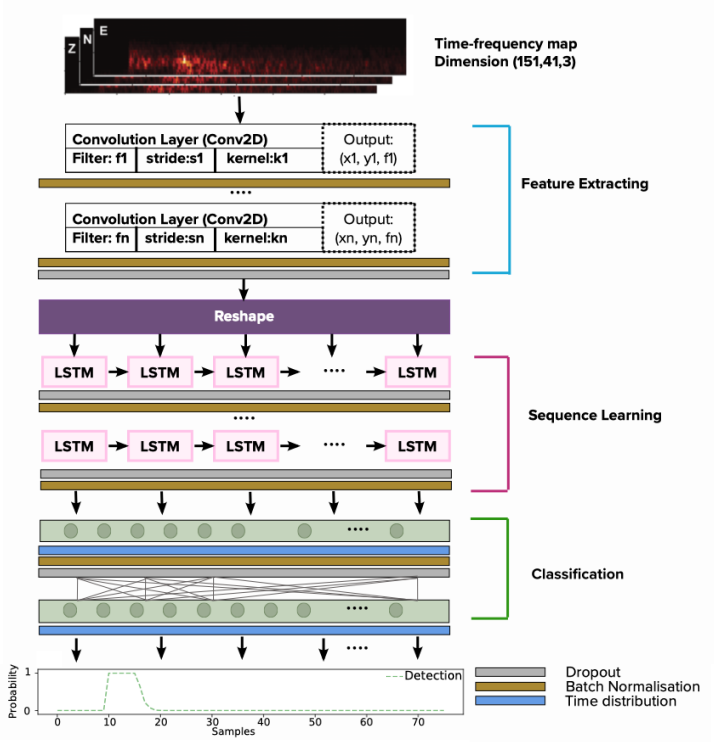


Fig. 6. The basic NN architecture of LightEQ-NN consists of three components. (1) CNN layers for feature extraction; (2) regular or residual LSTM blocks for sequence learning; and (3) classification layers.

upper threshold of 0.5 and a lower threshold of 0.25. If any value in the prediction vector exceeds the upper threshold, we mark the input as an earthquake. Once this condition is met, the classification remains active as long as subsequent values stay within the two thresholds. However, if no value ever crosses the upper threshold and all predictions stay below or between the thresholds, the input is classified as noise. This approach improves robustness by suppressing brief fluctuations in prediction values while preserving sensitivity to true events. The specific number of layers and their configurations are determined via neural architecture search and hyperparameter tuning.

4.3.2 Architecture and Hyperparameter Search. Building on the baseline architecture in Figure 6, we conduct an extensive NN architecture search and hyperparameter optimizations of 400 trials to identify a set of models with high accuracy and reduced loss. However, the design space for this architecture is vast and consists of many possible configurations, as shown in Table 3. Therefore, performing a brute-force search for the best possible parameters is infeasible. Alternatively, we employ Bayesian optimization using the Tree-structured Parzen Estimator (TPE) [4], which estimates the performance of hyperparameters based on historical evaluations and selects new hyperparameter configurations for subsequent testing. Further details on the architecture and hyperparameter search are available in the conference version of this article [61].

4.3.3 Selected Models. We select three models based on the architecture search and hyperparameter optimization. We base our selection on key metrics, including test accuracy, the number of parameters, and output dimensions, while considering constraints on RAM and flash consumption. The three selected models are described below:

Table 3. Hyperparameter Search Space for LightEQ

Design Dimension		Range	# of Choices
Number of filters		8, 16, 32	3
Kernel size		3, 5, 7, 9	4
Stride size		1, 2	2
Selection	Number of Conv. layers	1, 2, 3, 4, 5	5
	Number of Conv. blocks	3, 4, 5	3
Number of LSTM. layers		1, 2, 3	3
LSTM units		32, 64	2
CNN dropout		0.3–0.7	uniform
LSTM dropout		0.5–0.9	uniform
Batch size		50, 80, 100	3
Epochs		20, 30, 40	3

ModelSmall: This model begins with a CNN layer featuring a kernel size of 7×7, 8 filters, and a stride of 2, which reduces the output dimensions from 151 to 75. This is followed by a batch normalization layer, a ReLU activation layer, and a dropout layer with a rate of 0.31. Next, the model includes a single LSTM layer with 32 units, paired with a dropout layer with a rate of 0.61 to mitigate over-fitting. Finally, the model has two classification layers consisting of a dense layer with 64 units, followed by a dropout layer with a rate of 0.89, and a final dense layer with a single output unit per timestep, producing a probability vector that indicates the likelihood of earthquake activity at each sub-interval of the input..

ModelMedium: This model comprises five CNN residual blocks and a single LSTM layer. Each CNN block contains three CNN layers.[2] Following the CNN blocks is a dropout layer with a rate of 0.59, followed by an LSTM layer with 32 units. Another dropout layer, also with a rate of 0.59, is applied after the LSTM layer. Finally, the model includes a dense layer with 64 units and a dropout rate of 0.58, followed by a dense layer with a single output unit.

ModelLarge: This model features three CNN residual blocks and a single LSTM layer. Each residual block comprises three CNN layers, followed by a dropout layer with a rate of 0.4. The CNN blocks are succeeded by an LSTM layer with 32 units and a dropout layer with a rate of 0.73. The final layers of the model include a dense layer with 64 units, followed by a dropout layer with a rate of 0.66, and a final dense layer with a single output unit.

Output Dimension: ModelSmall and ModelLarge have an output dimension of 76x1, while ModelMedium has an output dimension of 38x1. These correspond to 76 and 38 sub-intervals per input segment, respectively. We define point-wise accuracy as the proportion of correctly predicted labels at each output timestep, i.e., whether each point in the prediction vector correctly identifies the presence or absence of earthquake activity. This metric reflects how well the model localizes events within the input window. We also compute an integrated binary classification metric, which we refer to as detection accuracy, based on whether the model correctly identifies the entire segment as either an earthquake or noise. While all three models achieve similar point-wise accuracy of 99.5%, their detection accuracy varies due to differences in output resolution. We further analyze this in Section 5.8.

4.4 System Design

This section discusses system and implementation details and outlines the memory requirements of each component.

[2]Detailed configurations of kernel size, stride, and number of filters can be found in the GitHub repository

Seismometer Readings. We receive seismic data over a serial connection, where each reading includes three 32-bit float amplitudes (12 bytes). To handle byte-alignment issues, we encode each reading using consistent overhead byte stuffing (COBS) [7], which adds two bytes, one for overhead and one for delimiting, bringing the total to 14 bytes per reading. Data arrives at a rate of 100 Hz. To reduce CPU overhead, we use an asynchronous UART with EasyDMA [38], which enables direct memory access with minimal CPU involvement. However, continuously handling data at 100 Hz prevents the system from entering low-power sleep modes. To mitigate this, we experiment with increasing the UART buffer size so that the CPU is interrupted only when the buffer is full, rather than on every incoming sample. While larger buffers reduce interrupt frequency and improve energy efficiency, they require additional memory and also add delays for data availability. For a UART buffer size of N, the memory requirement in bytes is:

$$\text{UART Buffer (bytes)} = N \times \text{COBS encoded reading size} = N \times 14.$$

Once the UART buffer is full, an interrupt is generated. The system then decodes each COBS-encoded message back into its original 12-byte form and copies the results to an application-level message queue. This message queue requires:

$$\text{Message Queue (bytes)} = N \times \text{COBS decoded reading size} = N \times 12.$$

STA/LTA. To evaluate detection latency and processing efficiency, we configure the UART buffer to store 1, 10, 100, or 1,000 readings, triggering STA/LTA updates every 10 msec, 100 msec, 1 sec, or 10 sec, respectively. These UART settings, however, do not affect the memory requirements of the pre-filtering stage. As shown in Figure 5, the pre-filter maintains three circular buffers for the LTA window. Because the STA window is contained within the LTA window, no additional memory is required for STA. Given that each sample is stored as a 32-bit float (4 bytes), the total buffer memory for the maximum LTA setting (4,000 samples) is:

$$\text{Prefilter Memory (bytes)} = \text{LTA length} \times \text{channel} \times \text{bytes per sample} = 4000 \times 3 \times 4 = 48,000.$$

When the STA/LTA algorithm flags an input as a potential earthquake, the trigger is driven by the most recent portion of the LTA buffer, i.e., the full STA window. In response, we construct a buffer by combining the flagged data with a continuation of incoming readings, resulting in 6,000 samples per channel. We then pass this complete sequence to the STFT component to generate a time-frequency map for NN inference. With three channels, 6,000 samples per channel, and 32-bit floating-point values, the total memory required for the buffer is approximately 70 kB.

STFT. Once we collect the 6,000 samples per channel, we pass them to the STFT function. The STFT transforms each channel's time-domain signal into a time-frequency representation, allowing the NN to analyze how frequency content changes over time. We configure the STFT with a window size of 80 and an overlap of 40, producing 151 time frames and 41 frequency bins per channel [37]. Since the STFT output exceeds the input buffer capacity, we reuse the buffer and extend it with an additional 2.24 kB to store the 191 surplus values per channel.

$$\begin{aligned}\text{STFT Buffer Memory (bytes)} &= (\text{Input buffer} + \text{Extra buffer}) \times \text{channel} \times \text{bytes per sample} \\ &= (6,000 + 191) \times 3 \times 4 = 74,292\end{aligned}$$

NN. We quantize the NNs to 8-bit integer weights and activations, while retaining floating-point inputs and outputs. These quantized models run on MCUs with as little as 100 kB of memory. To minimize memory overhead, we reuse the STFT buffer to store the network's output. The following equation accounts for all inputs, outputs, and intermediate results when measuring memory requirements during LightEQ pipeline execution. The memory required to run the NN is

accounted for separately in Section 5.8.

$$\begin{aligned}\text{Total Buffer Memory (bytes)} &= \text{ Prefilter Memory + STFT Buffer Memory} \\ &= 48,000 + 74,292 = 122,292\end{aligned}$$

4.5 Implementation

We implement LightEQ on the Zephyr RTOS [62] version 3.6 and train LightEQ-NN using TensorFlow [51]. For on-device inference, we deploy the model with LiteRT for MCU, previously known as TensorFlow Lite for MCUs [14]. To accelerate the entire pipeline, including preprocessing steps such as STFT and NN inference, we leverage the DSP capabilities of modern Cortex-M4/7/33 MCUs [32]. For NN execution, we integrate the CMSIS-NN library [25], which provides optimized kernels for ARM Cortex-M processors.

While LightEQ is platform-independent and not tied to a specific MCU, we evaluate on-device NN inference using two typical low-power SoCs: the nRF52840 (nRF52) [38] and the nRF5340 (nRF53)[39]. The nRF52 features a 64 MHz Cortex-M4 CPU with FPU, 1 MB of flash, 256 kB of RAM, and DSP instruction capabilities. The nRF53, by contrast, has a dual-core Arm Cortex-M33 with FPU, clockable at either 128 MHz or 64 MHz; in our configuration, we use the 64 MHz clock. It also provides 1 MB of flash, 512 kB of RAM, and DSP instruction capabilities. We evaluate NN inference performance on both MCUs, but we demonstrate the complete end-to-end pipeline exclusively on the smaller nRF52 platform to highlight LightEQ's efficiency on highly resource-constrained hardware.

5 Evaluation

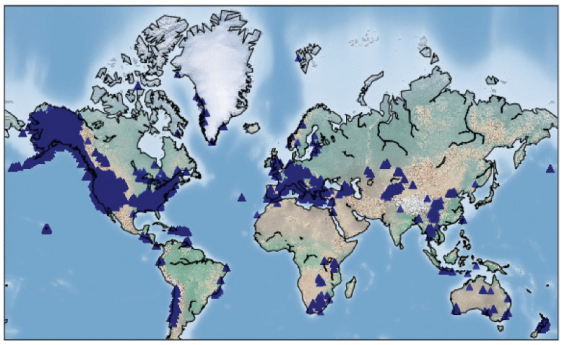
This section presents a comprehensive evaluation of LightEQ, analyzing both individual components and overall performance. We begin by describing the dataset and the evaluation metrics used for the evaluation. In Section 5.3, we evaluate several STA/LTA pre-filter configurations and select two configurations for further analysis. In Section 5.4, we analyze the memory requirements of the NNs, compare their performance against SOTA methods, and present results on a sample of input data. In Section 5.5, we evaluate the full pipeline performance, including the NN, using the two selected STA/LTA configurations. In Section 5.6, we examine the impact of UART interrupts on LightEQ's overall energy profile. In Section 5.7, we estimate the battery lifetime, followed by a discussion of key insights in Section 5.8.

5.1 Dataset and Data Engineering

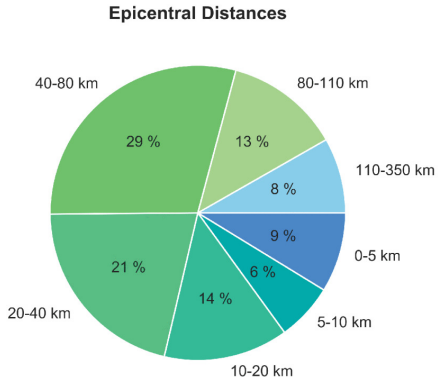
In the following, we introduce our training, validation, and test dataset, STEAD [35], which is a widely used seismological benchmark dataset and has been adopted by several prior studies [34, 37, 60, 63], enabling direct and reproducible comparisons across methods.

Dataset and Training. The dataset comprises 1,265,657 one-minute-long three-component seismograms sampled at 100 Hz, corresponding to 6,000 samples per component. Of these, 1,030,231 are labeled as earthquake recordings associated with events occurring between January 1984 and August 2018, and 235,426 are labeled as non-earthquake recordings. Each earthquake event is fully contained within a single one-minute recording. Importantly, earthquake recordings contain both event-related seismic signals and ambient background noise, whereas the non-earthquake class consists of recordings without earthquake events. Earthquake waveforms are recorded by 2,613 seismic stations distributed worldwide (see Figure 7(a)), with stations located within 350 km of the corresponding epicenters (see Figure 7(b)). The non-earthquake class includes both cultural (anthropogenic) and non-cultural seismic noise [34].

While our pipeline operates on a moving window and typically requires continuously sampled sensor data, we use one-minute windows per waveform to ensure a fair comparison with baseline



(a) Geographic distribution of seismic stations in the STEAD dataset that recorded earthquake waveforms (navy-blue triangles).



(b) Distribution of epicentral distances between earthquakes and recording stations in the STEAD dataset.

Fig. 7. Overview of the spatial coverage and epicentral distance characteristics of the STEAD dataset. (a) Global distribution of seismic stations that recorded earthquake events. (b) Distribution of epicentral distances between earthquakes and recording stations. Source: STEAD, Mousavi et al., licensed under CC BY 4.0 [36].

Table 4. Subsequences Used for Estimating the Standard Deviation of Gaussian Noise Padding. We Ignore the First and Last 100 Samples to Avoid Distortion from Edge Effects

Padding Side	Start Index (a)	Window Length (n)
Left (Start)	100	300
Right (End)	5,600	300

NN models. The dataset is split into training (85%), validation (5%), and test (10%) sets. Following the approach used in CRED [37], we apply data augmentation to the training set to improve model robustness. Augmentation includes superimposing a secondary earthquake signal at the empty parts of the trace, introducing random Gaussian noise levels, shifting seismic events, adding gaps in noise, and selectively dropping channels.

Continuous Data Simulation. A continuous data stream is necessary to simulate the real-world seismic monitoring. However, to our knowledge, no publicly available continuous labeled dataset exists for this purpose. Consequently, we generate synthetic continuous data by extending each STEAD waveform with predefined noise segments appended to both ends. For this padding, we use Gaussian noise that mirrors the waveform's statistical properties. We calculate the noise parameters using stable segments of the waveform from the start and end of the waveform as detailed in Table 4. The standard deviation (σ) for the Gaussian noise is estimated as

$$\sigma := \frac{\sqrt{\pi/2}}{n} \sum_{i=0}^{n-1} |wf[a+i]|,$$

where a is the subsequence start index and n is the window length. This ensures that the added noise reflects the background characteristics of the waveform and avoids abrupt amplitude changes at the stitching boundaries. We then use this synthetic continuous dataset to evaluate the complete LightEQ pipeline in Section 5.5.

5.2 Metrics

We evaluate our NN models using two complementary approaches: point-wise evaluation and event-level evaluation. Point-wise evaluation measures how precisely the model localizes earthquakes within the seismogram by treating each time step as an independent regression decision. This provides fine-grained feedback during training and enables the model to capture detailed temporal characteristics of seismic activity. In contrast, event-level evaluation assesses whether the model correctly identifies the presence of an earthquake anywhere within a recording of 60 seconds, without requiring exact localization in time. Throughout this article, we refer to event-level evaluation as detection accuracy, and we report precision, recall, and F_1 score based on this measure. To transition from point-wise outputs to event-level detection, the model predictions are post-processed using a thresholding function to produce binary classifications. A detection is considered correct if the model flags an earthquake at any point within the input window, even if the timing is slightly offset.

For event-level evaluation, we label each input seismogram as either an earthquake, if it contains at least one earthquake signal, or noise, if it contains only background noise. A true positive (TP) occurs when a seismogram containing an earthquake is correctly identified by the NN, regardless of whether it includes one or multiple earthquake signals. A true negative (TN) corresponds to a noise-only seismogram correctly classified as noise. Conversely, a false positive (FP) denotes a noise seismogram incorrectly classified as containing an earthquake, while a false negative (FN) refers to an earthquake-containing seismogram misclassified as noise. These definitions are consistent with standard evaluation protocols in seismic event detection [28, 34, 37, 63].

We calculate accuracy to measure the overall proportion of correct classifications:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Although accuracy provides a general sense of model performance, it may be misleading in imbalanced datasets. To address this, we also compute precision and recall. We use precision to measure how often the model is correct when it predicts an earthquake:

$$precision = \frac{TP}{TP + FP}.$$

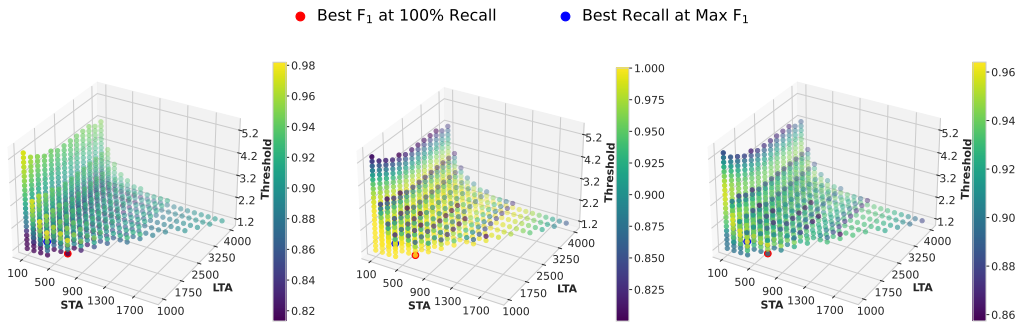
We compute recall to assess the model's ability to detect actual earthquake events:

$$recall = \frac{TP}{TP + FN}.$$

The F_1 score summarizes the tradeoff between precision and recall as their harmonic mean:

$$F_1 = \frac{2 * precision * recall}{precision + recall}.$$

The choice of precision, recall, and F_1 score reflects the operational goals of on-device seismic event detection. In earthquake engineering applications, missed detections (FN) and false alarms (FP) have asymmetric consequences depending on the deployment scenario. For early warning and rapid alerting systems, high recall is often prioritized to minimize missed events, even at the cost of occasional false alarms. In contrast, for long-term monitoring and distributed sensing deployments, balanced performance is required to avoid excessive false alerts and unnecessary data transmission. Reporting precision and recall separately, together with the F_1 score as a summary measure, allows performance to be interpreted according to these differing operational priorities.



(a) Precision of STA/LTA (b) Recall of STA/LTA (c) F_1 score of STA/LTA

Fig. 8. STA/LTA grid search results showing precision, recall, and F_1 score across different STA lengths, LTA lengths, and threshold configurations. The highlighted configurations correspond to settings selected for further evaluation in LightEQ. We include only configurations where all metrics exceeded 80%.

Table 5. Comparison of STA/LTA Performance Under Two Configurations. The First Corresponds to the Setting with the Highest F_1 Score Among those Achieving 100% Recall, While the Second Corresponds to the Setting with the Highest Recall Among those Achieving the Maximum F_1 Score

Metric	**STA/LTA Configurations**	
	Best F_1 at 100% Recall (LTA = 1,250, STA = 600, Threshold = 1.2)	Best Recall at Max F_1 (LTA = 1,000, STA = 400, Threshold = 1.8)
Precision	82%	92%
Recall	100%	98%
F_1 Score	90%	95%

5.3 STA/LTA Pre-Filter

We start our evaluation with the STA/LTA component of LightEQ, exploring different STA and LTA window lengths and threshold values (see Table 2). We assess performance using precision, recall, and F_1 scores on the test split of the STEAD dataset. Figure 8 illustrates how performance varies across the STA/LTA parameter space. For visual clarity, we plot only those configurations where precision, recall, and F_1 score are each greater than or equal to 80%. We observe that STA/LTA is highly sensitive to threshold changes: small increases in the threshold generally improve precision at the expense of recall. Additionally, the choice of STA and LTA window sizes influences temporal resolution and noise suppression [53]. While longer LTA windows improve stability by reducing sensitivity to noise, they can also suppress valid detections. Similarly, longer STA windows generalize sharp transients by averaging them with surrounding noise. While this can reduce noise-induced FPs, it also diminishes sensitivity to abrupt seismic onsets.

Based on this analysis, we select two STA/LTA configurations for further evaluation. These two configurations reflect distinct operational priorities (see Table 5). The first configuration achieves a recall of 100% and the best F_1 score among all tested settings. It is designed to maximize recall and prioritize detecting as many earthquake events as possible while still maintaining a strong F_1 score. In practice, this setting filters out only 3% of noise samples, making it appropriate for scenarios where failing to detect an event is more costly than issuing a false alarm, such as real-time monitoring. In contrast, the second configuration provides a more balanced tradeoff between sensitivity and precision. While it achieves an optimal F_1 score and nearly perfect recall, missing only 1.4% of earthquake events, it substantially improves noise rejection by filtering out 74% of

Table 6. Resource Requirements and Inference Time for 8-bit Integer and 32-bit Float Models with CMSIS-NN Enabled for LightEQ-NN. ModelLarge is not Deployable on the nRF52840 MCU. "LightEQ updated" Refers to Updated Results Obtained using the TFLite Micro Optimizations [3] with Zephyr Version 3.6.0, While "LightEQ" Corresponds to the Values Reported in Our LightEQ Conference Submission [61]

			LightEQ updated			LightEQ [61]			improvement (over int8)
			int8	float	improvement (int8 vs. float)	int8	float	improvement (int8 vs. float)	
nRF52840	ModelSmall	Exe (sec)	0.4	5.5	13.7x	0.9	-	-	2.2x
		RAM (kB)	94.3	124.9	1.3x	193.5	-	-	2.0x
		Flash (kB)	186.9	269.9	1.4x	583.9	-	-	3.1x
	ModelMedium	Exe (sec)	2.3	39.9	17.3x	3.7	-	-	1.6x
		RAM (kB)	99.1	155.4	1.6x	147.9	-	-	1.5x
		Flash (kB)	243.2	447.1	1.8x	448.8	-	-	1.8x
nRF5340	ModelSmall	Exe (sec)	0.3	0.9	3.0x	0.8	3.6	4.5x	2.7x
		RAM (kB)	94.3	124.9	1.3x	193.5	224.9	1.2x	2.0x
		Flash (kB)	187.4	270.1	1.4x	596.1	490.3	-1.2x	3.2x
	ModelMedium	Exe (sec)	1.8	7.4	4.1x	3.5	47.6	13.6x	2.0x
		RAM (kB)	99.1	155.4	1.6x	147.9	205.3	1.4x	1.49x
		Flash (kB)	253.7	447.6	1.8x	451.0	592.3	1.3x	1.9x
	ModelLarge	Exe (sec)	10.8	-	-	20.3	-	-	1.9x
		RAM (kB)	298.7	-	-	398	-	-	1.3x
		Flash (kB)	315.9	-	-	725.5	-	-	2.3x

noise data. This makes it better suited for automated cataloging or post-processing workflows, where excessive FPs can result in unnecessary computational and transmission overhead. We use both configurations to evaluate overall pipeline accuracy, but in Section 5.6 and Section 5.7 we focus on the high-recall setting to assess battery lifetime.

5.4 Neural Network

This section evaluates the effectiveness of the standalone LightEQ-NN. We begin by analyzing the resource requirements of the proposed models. This is followed by a comparative evaluation of their performance, measured in terms of precision, recall, and F_1 score, against other SOTA models. We include both 8-bit integer and 32-bit floating-point versions in the comparison. We also analyze parameter counts and assess the deployability of our models and the SOTA baselines on MCUs. Finally, we illustrate the output of our NN model with visual examples.

5.4.1 Resource Requirements. Unlike our prior work [61], which relied on unrolled LSTMs, LiteRT for MCU (commit 9245002, Nov 6, 2024) eliminates this requirement, reducing execution time as well as RAM and flash usage (Table 6). This table summarizes the execution time, RAM, and flash consumption for all three LightEQ-NN on the nRF52840 and nRF5340 MCUs. ModelLarge is only deployable on nRF5380. In the conference paper [61], we were limited to deploy only the integer-quantized versions of ModelSmall and ModelMedium on the nRF52840. The integer-quantized versions had relatively high memory usage, which prevented us from deploying the floating-point models. However, recent optimizations of LiteRT for MCU have significantly improved resource efficiency. As a result, we can now deploy the floating-point models successfully and have also observed improved performance in the integer-quantized models. For instance, the integer-quantized version of ModelSmall now consumes approximately half the memory and flash compared to our previous work; we observe similar reductions for ModelMedium as well. These optimizations not only reduce memory consumption but also lead to substantial improvements

Table 7. LightEQ-NN and SOTA Models. The Table Lists Each Model's Input and Output Dimensions, Parameter Count (Params), and Deployability on MCUs. Among the Models Compared, Only STA/LTA, SeismicSense, and LightEQ-NN are Deployable Directly on MCUs. The SOTA Models EQTransformer and LCANet Perform both Earthquake Detection (Det.) and P-/S-wave Phase Picking (Ph.), Whereas Phasenet and Picknet Focus Solely on Wave Identification but can be adapted for Detection Tasks. Although LEQNet has only 40k Parameters, its Deployment is Constrained to Edge Devices that Provide at Least 1 MB of RAM. All Numbers, Except those for LightEQ-NN, are Taken Directly from LCANet; for LightEQ-NN, we Report Model Sizes of Integer-Quantized Models. The Lower Portion of the Table Lists Models that Address Tasks Beyond those of LightEQ-NN

Model		Input Size	Output Size	Params	Model Size	MCU Deployable	Task	
							Det.	Ph.
STA/LTA [30]		$1,250 \times 3$	1×3	3	-	✓	✓	
CRED [37]		$151 \times 41 \times 3$	38×1	293k	3.7 MB		✓	
LightEQ	ModelSmall	$151 \times 41 \times 3$	76×1	29k	0.09 MB	✓	✓	
	ModelMedium	$151 \times 41 \times 3$	38×1	73k	0.10 MB	✓	✓	
	ModelLarge	$151 \times 41 \times 3$	76×1	151k	0.29 MB	✓	✓	
Phasenet [64]		$6,000 \times 3$	$6,000 \times 3$	269k	3.2 MB			✓
Picknet [56]		$6,000 \times 3$	$6,000 \times 2$	6,044k	72 MB			✓
EQTransformer [34]		$6,000 \times 3$	$6,000 \times 1$	378k	5.2 MB		✓	✓
LCANet [63]		$6,000 \times 3$	$6,000 \times 3$	210k	3.7 MB		✓	✓
LEQNet [28]		$6,000 \times 3$	$6,000 \times 3$	40k	0.94 MB		✓	✓
SeismicSense [60]		$6,000 \times 3$	$6,000 \times 3$	169k	0.18 MB	✓	✓	✓

in execution time. Furthermore, leveraging CMSIS-NN significantly accelerates integer-quantized models: with CMSIS-NN, ModelSmall now executes in just 0.4 seconds and ModelMedium now runs in 2.3 seconds, representing at least a 1.6-fold improvement over the conference paper results. When comparing the execution time of integer-quantized models to their floating-point counterparts, we observe a minimum 13-fold improvement. However, on nRF5380, the integer-quantized models with CMSIS-NN now run twice as fast compared to previous work, while the floating-point models show at least a 4-fold improvement.

5.4.2 Comparison with Other Models. In this section, we compare the performance of our models with SOTA approaches. Table 7 highlights a comparative analysis between LightEQ and SOTA models, including STA/LTA, CRED, Phasenet, Picknet, EQTransformer, LCANet, LEQNet and SeismicSense. These models differ in input/output dimensions, parameter counts, and functional capabilities. For STA/LTA, we use a configuration that achieves perfect recall from Section 5.3. The remaining models feature more complex architectures with parameter counts ranging from 40k to 6,044k, which increases their computational demands and renders them unsuitable for MCU deployment. LEQNet contains 40k parameters, but it still requires edge devices with at least 1 MB of RAM due to its use of transformer and attention mechanisms. SeismicSense next to LightEQ is the only model deployable on an MCU, with a model size of 189 kB. Additionally, SeismicSense performs phase picking, making it a strong candidate for MCU deployment. As shown in Figure 9, SeismicSense achieves the same 99% recall as ModelSmall_{int}, and has a higher precision. In contrast, ModelSmall_{int} is half the size of SeismicSense and executes five times faster.

We also compare the precision, recall, and F_1 scores for all three LightEQ-NN (both float and integer-quantized) models with other SOTA models. Since Phasenet and Picknet are not originally designed for event detection but rather for P- and S-wave picking, we approximate detection performance by using their best F_1 score for P- and S-wave identification. All three LightEQ-NN models achieve high precision, recall, and F_1 scores. EQTransformer and LCANet reach the

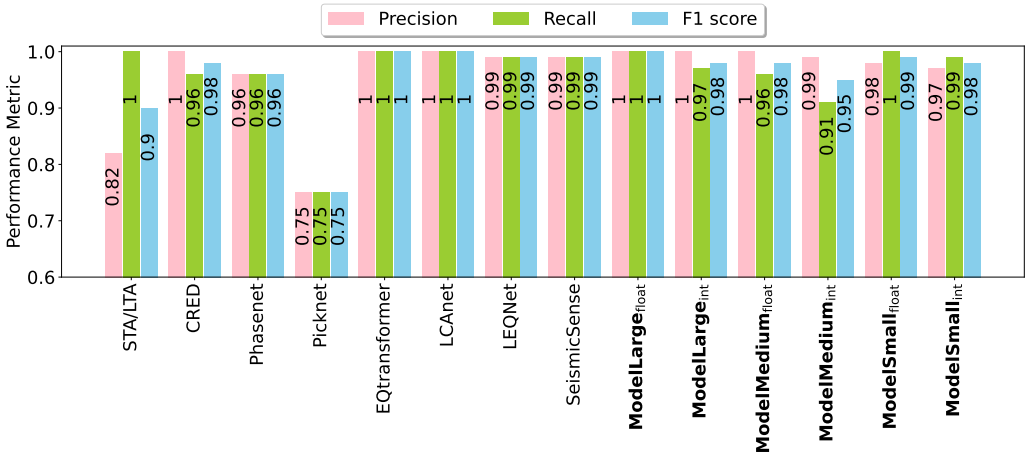


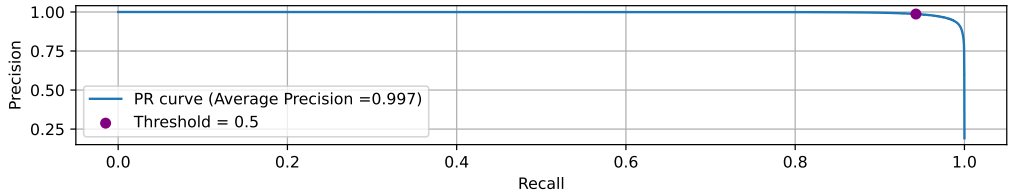
Fig. 9. LightEQ-NN compared to SOTA models: STA/LTA refers to a non-deep learning model, while CRED, Phasenet, Picknet, EQTransformer, LCANet, LEQNet, and SeismicSense are NN models. ModelLarge, ModelMedium and ModelSmall are float and integer quantized versions of LightEQ-NN models. Overall, LightEQ-NN achieves performance comparable to CRED and slightly lower than EQTransformer, LCANet, LEQNet, and SeismicSense. However, it offers significantly higher resource efficiency, making it more suitable for deployment on constrained devices with only 100 kB of RAM. The remainder of the on-device evaluation uses ModelSmall$_{int}$ as the representative model.

highest overall performance with an F1 score of 100%, closely followed by CRED at 98%. The LightEQ ModelLarge$_{float}$ model performs on par with EQTransformer and LCANet, also achieving a perfect F_1 score. Quantizing this model leads to a slight performance drop of 2% in F_1, leaving the quantized version still comparable to CRED. Although CRED requires four times more parameters than ModelMedium and ten times more than ModelSmall, LightEQ-NN delivers similar accuracy. SeismicSense slightly outperforms ModelSmall*int* in F1 while maintaining a comparable recall of 99%, but ModelSmall*int* remains five times more resource-efficient.

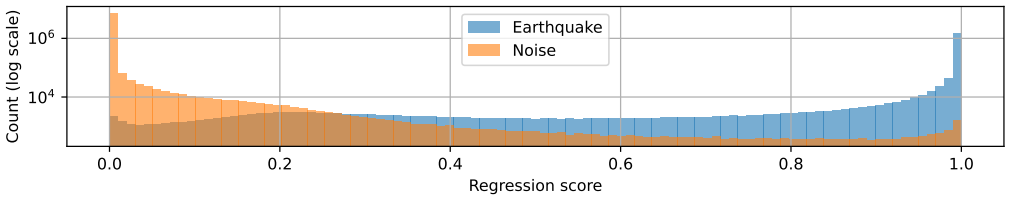
Based on these results, ModelSmall outperforms ModelMedium while using fewer parameters, highlighting that a larger model does not necessarily lead to better performance. On the nRF53 platform, where both ModelLarge$_{int}$ and ModelSmall$_{float}$ can be deployed, ModelSmall$_{float}$ offers a better tradeoff, using fewer resources while achieving superior F1 performance. This demonstrates the effectiveness of efficient model design in resource-constrained environments. Therefore, for the remainder of our on-device evaluation, which focuses on LightEQ's long-term deployment, we use ModelSmall$_{int}$ as the representative model.

We evaluate performance robustness beyond a fixed threshold using a precision–recall analysis of the NN output from ModelSmall$_{float}$ (see Figure 10(a)). The curve shows the tradeoff between detection probability and false-alarm rate as the confidence threshold varies. Together with the score distributions in Figure 10(b), the analysis shows that residual false detections arise from a small number of noise samples that are classified as earthquakes with high confidence.

5.4.3 Example Results. In this section, we present example results from LightEQ-NN. Figure 11 shows four seismograms with event detection outputs from ModelSmall$_{int}$. Each 1-minute input is mapped to a probability vector of 76 points. Figures (a)–(c) illustrate detections for three different earthquakes, while Figure (d) shows a recording of background noise. Each of the seismic signals was recorded with three components (Z, N, and E), as discussed in Section 2. Once passed through STFT preprocessing, the output is fed to the NN part. The earthquake magnitudes, source-receiver dis-



(a) Precision–recall (PR) curve. The curve illustrates the trade-off between detection probability (recall) and false-alarm rate (precision) across all possible regression thresholds. The high average precision (AP = 0.997) indicates near-perfect ranking of earthquake and noise samples, while the curve shape highlights how precision degrades as progressively lower-confidence detections are accepted. The threshold used in this article is 0.5.



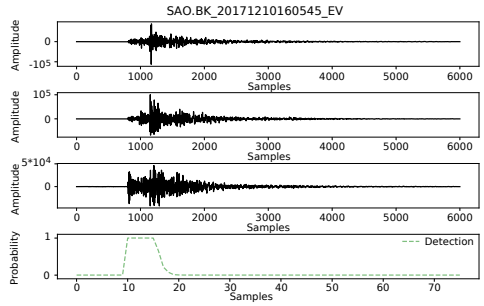
(b) Distribution of per-timestep regression scores for earthquake and noise. The distribution shows that the model assigns scores near zero to most noise timesteps and scores near one to most earthquake timesteps, with a limited region of overlap. A small high-score noise tail persists and contributes to residual false alarms.

Fig. 10. Precision–recall analysis and NN score distributions. (a) presents the precision–recall curve of the NN outputs, and (b) presents the associated per-timestep score distributions for earthquake and noise samples. Together, these results indicate that the near-rectangular precision–recall curve reflects strong score separation and confident predictions rather than threshold sensitivity.

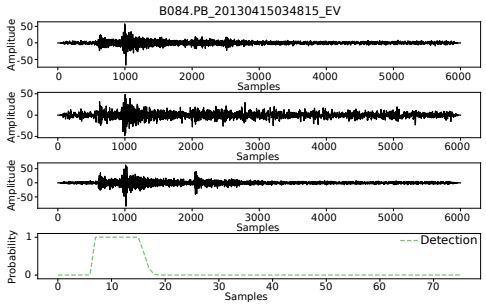
tances, and SNR vary between those examples, and the categorization difficulty increases from (a) to (d). While the waveforms of the magnitude 2.43 earthquake in (a) and the magnitude 0.5 earthquake in (b) look similar at first sight, (b) has a significantly lower amplitude, making the signal less impulsive and thus more difficult to classify than (a). The magnitude 5.7 of the earthquake in (c) is recorded at a greater source-receiver distance, making it appear noisier. Figure (d) is a recording of background noise that is very similar to (c), but still LightEQ-NN model successfully classifies it as noise.

5.5 Pipeline Accuracy

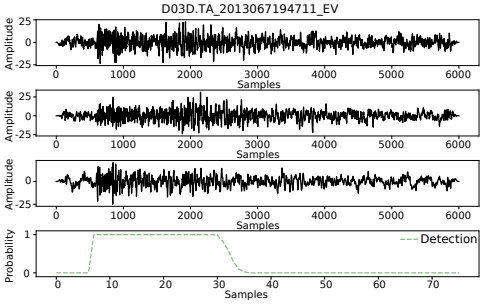
In Section 5.3, we evaluated the STA/LTA pre-filter in terms of threshold selection and window lengths. In this section, we extend the analysis to the complete LightEQ pipeline, examining how different STA/LTA configurations shape overall performance and the tradeoffs they introduce between noise rejection and event detection accuracy. Figure 12 presents a comprehensive overview of how precision, recall, and F_1 score vary across different STA/LTA parameter configurations when used as a pre-filter for the NN. While several high-threshold configurations appear to yield superior scores in the pipeline evaluation, our earlier analysis (see Figure 8) reveals that these settings often fail to suppress noise effectively. In such a setting, the STA/LTA filter passes most of the input to the NN, and the observed performance reflects the NN's capabilities on essentially unfiltered data rather than any meaningful contribution from the STA/LTA component. In Figure 8, we exclude plotting configurations where the standalone STA/LTA component yields precision, recall, or F_1 scores below 80%. In Figure 12, we focus on configurations with threshold values between 1.2 and 1.8, which strike a balance between effective filtering and meaningful contribution to the overall pipeline.



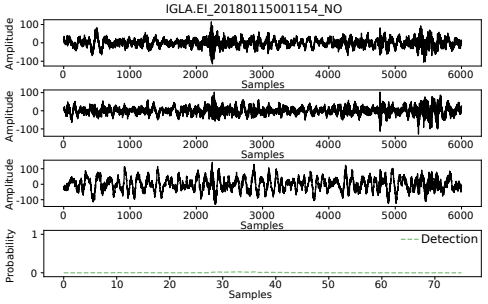
(a) Earthquake of magnitude 2.43 at a distance of 25.5 km



(b) Earthquake of magnitude 0.5 at a distance of 28 km

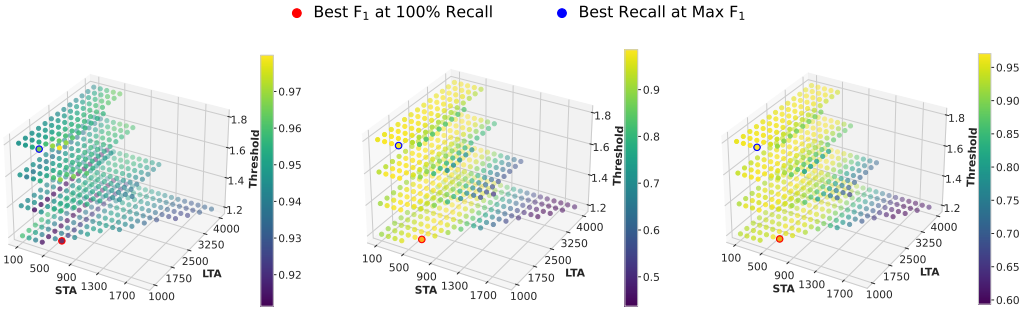


(c) Earthquake of magnitude 5.7 at a distance of 70 km



(d) Noise

Fig. 11. Visualization of selected results for LightEQ's ModelSmall$_{int}$ on the test set: (a) a seismogram with an earthquake with a magnitude of 2.43, a seismic source distance of 25.5 km, and an SNR of 50; (b) a seismogram with an earthquake with a magnitude of 0.5, a seismic source distance of 28 km, and an SNR of 15.9; (c) a seismogram with an earthquake with a magnitude of 5.7, a seismic source distance of 70 km, and an SNR of 104; and (d) a seismogram with noise only. We note that our model classifies all four input seismograms correctly. Visualizations follow Mousavi et al. [37].

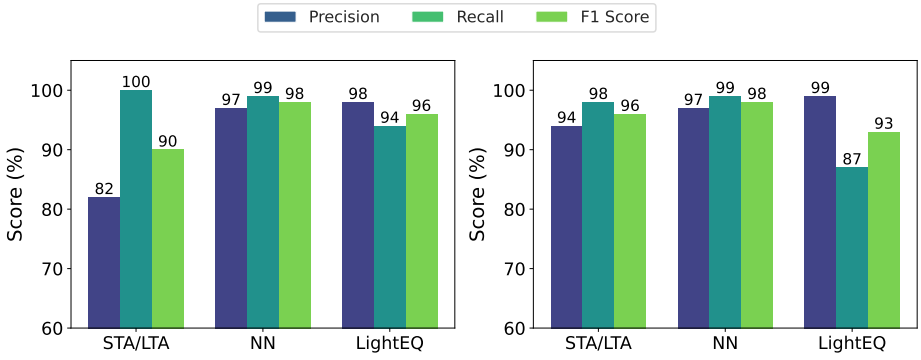


(a) Precision of LightEQ-NN when followed by STA/LTA

(b) Recall of LightEQ-NN when followed by STA/LTA

(c) F_1 score of LightEQ-NN when followed by STA/LTA

Fig. 12. Precision, recall, and F_1 score for ModelSmall$_{int}$ across different STA, LTA, and threshold configurations. The highlighted configurations (red and blue) correspond to selected configurations mentioned in Section 5.3. We include only the configurations where STA/LTA precision, recall, and F_1 score are all above 80%, and the threshold is between 1.2 and 1.8.



(a) Performance of the full detection pipeline using the STA/LTA configuration (LTA = 1250, STA = 600, Threshold = 1.2) followed by ModelSmall_{int}.

(b) Performance of the full detection pipeline using the STA/LTA configuration (LTA = 1000, STA = 400, Threshold = 1.8) followed by ModelSmall_{int}.

Fig. 13. Precision, Recall, and F_1 Score on the STEAD test dataset for LightEQ. In both subfigures, the first and second groups of bars show the standalone performance of the STA/LTA and ModelSmall_{int}. The final group represents the NN's performance when preceded by STA/LTA filtering. The slight drop in NN performance after STA/LTA filtering simply shows the network focusing on harder-to-detect events, which is essential for reducing false alarms in real deployments.

Figure 13 takes a closer look at our two selected STA/LTA configurations from Section 5.3. While the NN performs strongly in isolation (Figure 13), adding STA/LTA pre-filtering alters the event-to-noise ratio in the input. By removing obvious noise, the filter leaves the network with fewer but more ambiguous cases to classify, which leads to an apparent reduction in measured performance. This effect is evident in Figure 13(b), where filtering out 74% of the noise results in a drop in F_1 score from 98% to 93%. In contrast, when STA/LTA removes only 3% of the noise (Figure 13(a)), the performance drop is limited to just 2%. Importantly, this slight degradation is not a drawback. Instead, it reflects the network's ability to focus on subtle, difficult-to-detect events, an essential capability in practical scenarios where reducing false alarms is critical, especially under low-bandwidth constraints that limit the amount of data that needs to be sent to the cloud.

5.6 UART Buffering for Low-Power Operation

Deploying an unoptimized system in remote areas requires frequent battery replacements, which is impractical and significantly increases maintenance overhead. To reduce energy consumption, we optimize the entire data processing pipeline, leveraging DSP instructions and a recursive STA/LTA algorithm for efficient event detection. Running the pipeline on every incoming sample at 100 Hz forces the CPU to wake up frequently and substantially increases power consumption. In this section, we experimentally evaluate the impact of UART data buffering on overall energy consumption. Larger buffers reduce interrupt frequency by batching multiple measurements per wake-up, allowing the CPU to remain in sleep mode longer. Conversely, small buffers, such as size 1, trigger an interrupt on every sample and maximize CPU wake-ups. At our 100 Hz sampling rate, this corresponds to 100 interrupts per second for a buffer of size 1, compared to only 10 or 1 per second for buffer sizes of 10 or 100, respectively:

(a) Buffer size 1: Processes each sample immediately (100 Hz wake-up rate).
(b) Buffer size 10: Processes 10 samples at a time (10 Hz wake-up rate).
(c) Buffer size 100 or more: Wake-up rate drops to 1 Hz or less.

Table 8. Impact of UART Buffer Size on Current Consumption, Wake-Up Frequency, and Memory Requirements During Continuous Sampling and STA/LTA Processing at 3.3 V. Increasing the Buffer Size Decreases Energy Consumption Per Sample by Reducing How Often the MCU Wakes Up to Handle Incoming Data, At the Cost of Higher Memory Usage

UART Buffer Size (samples)	Processing Rate (times/sec)	Memory per Buffer (bytes)	Average Current (mA)	Energy per Sample (mJ)
1	100	14	2.3	0.073
10	10	140	1.8	0.059
100	1	1,400	0.6	0.020
1,000	0.1	14,000	0.6	0.018

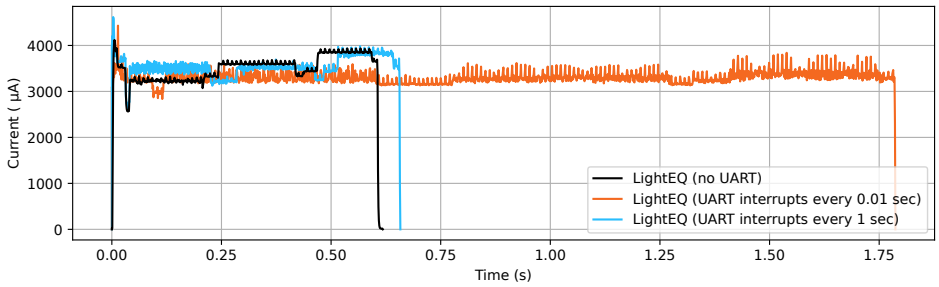


Fig. 14. Impact of UART interrupts on STFT and NN inference time for LightEQ. Frequent interrupts significantly increase execution time. Larger UART buffers reduce interrupt frequency, though a small overhead (0.04 sec) remains due to active DMA. The minimum execution time (0.62 sec) is measured with both UART and DMA disabled. For comparison, SeismicSense, without interrupts, takes nearly five times as long as LightEQ due to higher computational complexity.

Table 8 summarizes how buffer size affects energy consumption. Larger buffers significantly reduce energy per sample by lowering CPU wake-up frequency. However, this comes at the cost of increased memory usage and added latency, which can reduce system responsiveness. For instance, once the STA/LTA component triggers, the system must collect 5,250 samples for STFT processing. With a buffer size of 1, this acquisition takes 52.5 sec. In contrast, a buffer size of 1,000 samples introduces an additional 10-second delay while filling, extending the total acquisition time to 62.5 sec.

The effect of UART buffering is also evident in Figure 14. Smaller buffers trigger more frequent interrupts, leading to an increased CPU wake-up rate and longer total execution times for STFT and NN inference. In the best case, with no interrupts, STFT and NN execution complete in approximately 0.6 seconds. In contrast, with UART interrupts at 100 Hz, the same computation takes around 1.75 seconds, which is nearly three times longer. Although we have not directly tested UART interrupt effects on the SeismicSense model, its baseline execution time without interrupts is about 3 seconds. We expect that, similar to LightEQ, this would also lead to longer execution time.

5.7 Deployment Scenario and Battery Lifetime Estimation

In this section, we estimate the expected operational lifespan of LightEQ in a typical deployment scenario. The following analysis considers sensor configurations based on short-period geophones or MEMS-based sensors [10, 42, 45], where sensing hardware is passive and where computation and communication dominate the node-level energy budget. The system consists of nRF52840 MCU, which performs on-device signal processing and inference. Event detections are transmitted to

Table 9. Average Current and Daily Energy Consumption for Constantly Running STA/LTA with Buffer Updates and Some System-Level Operations, Such as Interrupt Handling, DMA, Power Management, and Periodic Sleep

Component	Processing Rate (times/sec)	Average Current (mA)	Daily Energy Consumption (mWh)
STA/LTA Execution and System-Level Operations	1	0.6	47.52
	100	2.3	182.16

Table 10. Execution Time, Triggers per Day, Average Current, and Energy Consumption for Each System Component after a STA/LTA Triggers. We assume that STA/LTA Triggers Occur Four Times Per Day, with Only One Representing a True Positive Event, Resulting in a Single LoRa Transmission

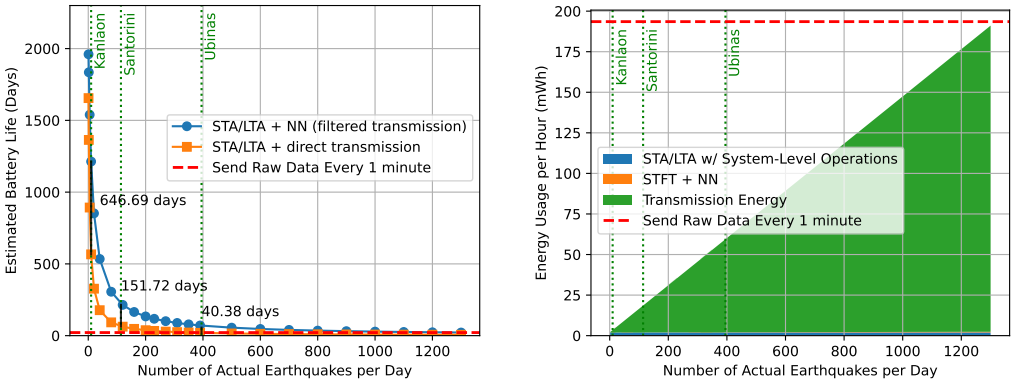
Component	Execution Time (sec)	Triggers per day	Duration (s/day)	Average Current (mA)	Daily Energy Consumption (mWh)
STFT	0.22	4	0.92	3.5	0.0028
LightEQ-NN	0.4	4	1.72	3.5	0.0051
LoRa TX (1-minute data)	40	1	40	29	3.2255

the cloud via LoRa communication configured with spreading factor 7 (SF7), corresponding to a bit rate of 5.47 kbps. Power is supplied by a 200 Wh lithium-ion battery,[3] of which only 50% is practically usable due to self-discharge and cutoff voltage constraints. This results in an effective energy budget of 100 Wh for on-device operation and communication.

To estimate the system's lifespan, we analyze the energy consumption of each system component. The STA/LTA algorithm runs continuously, along with buffer management and essential system-level functions, including interrupt handling, DMA, sleep, and power management. With a UART buffer size of 100 measurements, the system processes data at 1 Hz, consuming an average of 0.6 mA at 3.3 V. This results in a daily energy consumption of 47.52 mWh (see Table 9). Once the STA/LTA is triggered, the remaining pipeline is executed, which includes STFT, NN, and transmission of events using LoRa classified as earthquakes by the NN (see Table 10). The STFT and NN processes consume an average of 3.5 mA for 0.22 seconds and 0.4 seconds, respectively. When the NN classifies an input as an earthquake, the system transmits data via LoRa at spreading factor 7 (SF7) [11], consuming approximately 3.2255 mWh per transmission. This estimate assumes a payload corresponding to 60 seconds of sampled data (60 sec × 100 samples/s × 3 channels x 4 bytes/sample = 72,000 bytes), segmented into multiple LoRa packets. We assume the optimal case without retransmissions or acknowledgments.

We assume that the STA/LTA detector triggers four times per day, with each trigger initiating STFT preprocessing and NN inference, but only one of these corresponds to a TP that requires transmission. This estimate is based on a conservative scenario; however, in practice, the system may observe up to 20 triggers daily, with only one typically resulting in data transmission. Assuming a usable battery capacity of 100,000 mWh, the system is expected to operate for approximately 1,970 days. However, when the UART buffer size is reduced to 1, the daily energy consumption increases to 182.16 mWh (see Table 9), reducing the expected system lifetime to approximately 539 days. The energy contribution of the STFT and NN, triggered only a few times per hour, is minimal compared to the continuous STA/LTA operations. The energy overhead of these triggered computations has a negligible impact on the overall system power budget, effectively making them "free" to a large extent. This is evident in the following section.

[3]A typical 14.8 V, 13.2 Ah lithium-ion pack provides approximately 195 Wh of energy [50].



(a) Estimated lifetime of LightEQ versus two baselines: sending 1-minute of raw data every minute, sending 1-minute data after every STA/LTA trigger. Green dotted lines indicate average earthquake rates per day at three active volcanoes: Kanlaon (10/day) [2], Santorini (114/day) [20], and Ubinas (395/day) [17].

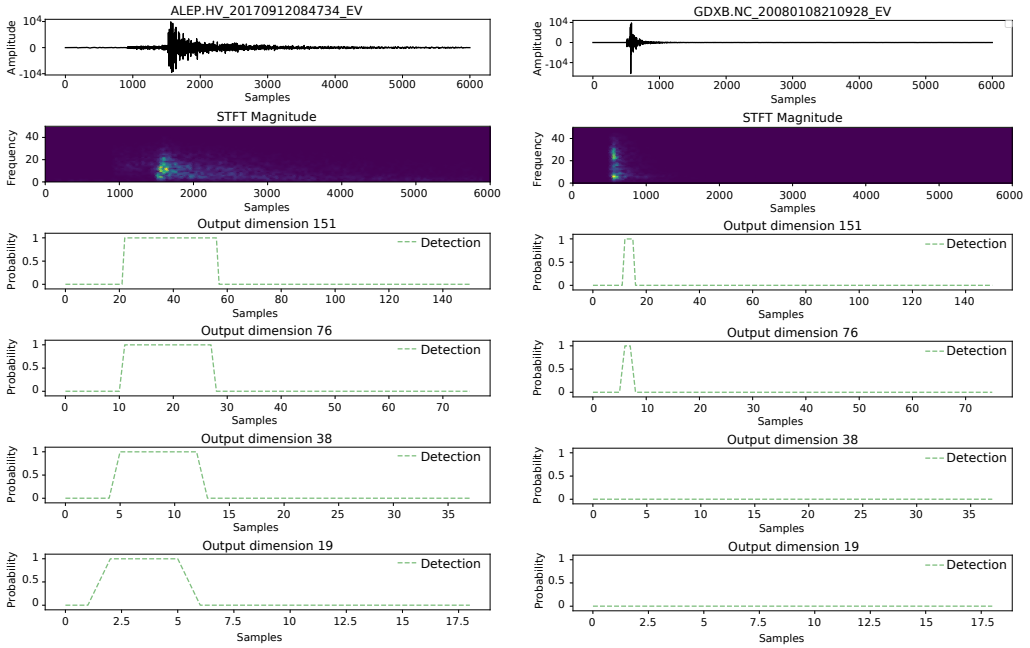
(b) Breakdown of LightEQ's hourly energy usage by component as a function of daily earthquake event rate. Energy for STA/LTA, along with system-level operations, remains constant, while STFT, neural network inference, and transmission energy increase with event frequency.

Fig. 15. Estimated battery life and corresponding energy consumption breakdown across varying earthquake event frequencies, with a processing frequency of 1 Hz.

5.7.1 Event Frequency and Its Impact on System Sustainability. This section examines how event frequency and different filtering strategies, from direct transmission to STA/LTA detection and STA/LTA with NN classification, affect overall energy consumption and battery life.

Figure 15(a) shows the estimated system lifetime as a function of the number of actual earthquake events per day. The green dotted line represents the average daily number of earthquakes across three active volcanoes. On average, about 10 earthquakes per day were observed at Kanlaon Volcano following its June 2024 eruption [2], 114 earthquakes per day at Santorini from January 2024 to March 2025 [20], and 395 earthquakes per day at Ubinas Volcano during its July 2019 eruptive unrest [17]. In the naive approach, the system transmits all raw data to the cloud in 1-minute packets, lasting only about 21 days on a 100 Wh energy reserve. In comparison, by using STA/LTA filtering at a 1 Hz processing rate and transmitting only potential earthquake events, the system can last up to 600 days under low seismic activity. However, in real deployments, where FPs are typically much more frequent, the number of transmissions increases accordingly, rapidly depleting the battery. In contrast, LightEQ adds an additional NN-based filter to suppress FPs and reduce unnecessary transmissions. This step is critical, as communication remains the most energy-intensive part of the system. At around 400 earthquakes per day, the energy cost of transmitting STA/LTA events matches that of the naive continuous-transmitting approach, reducing battery life to 21 days. Under the same conditions, LightEQ lasts 40 days and continues to outperform the naive baseline up to roughly 1,300 events per day.

Figure 15(b) breaks down the hourly energy consumption across LightEQ's processing pipeline. The STA/LTA, along with system-level operations, which run continuously, accounts for the dominant share of baseline energy use, exceeding the cost of NN inference. However, by applying additional filtering through NN, LightEQ successfully eliminates most false triggers before they lead to communication. Even as seismic activity increases, the energy spent on local computation remains minimal compared to the cost of wireless transmission. By dedicating a small amount of energy to on-device inference, LightEQ achieves a substantial reduction in communication



(a) A seismogram for a longer earthquake with a source-receiver distance of 34.81 km and event duration of about 20 seconds. Independent of the output dimensions, the earthquake is detected in all configurations.

(b) A seismogram for a comparably shorter earthquake with a source-receiver distance of 2.05 km and event duration of 4 seconds. The earthquake signal is not detected by the NN model with output of dimensions 38 and 19.

Fig. 16. NN with output dimensions of (151×1), (76×1), (38×1), and (19×1) for two seismograms. We only show one channel from each seismogram, although the NNs take three-channel seismograms as input.

overhead, resulting in a far more energy-efficient system under high-trigger conditions. The red dashed line in Figure 15(b) represents the energy consumption when sending raw data every minute.

5.8 Discussion and Limitations of LightEQ

In the following, we reflect on the results and limitations of LightEQ.

LightEQ-NN's output dimensions: In Section 5.4, we observe that ModelSmall and ModelLarge outperform ModelMedium, even though ModelMedium has more parameters than ModelSmall. Notably, despite having the fewest parameters, ModelSmall achieves accuracy very close to that of ModelLarge. Using CNN layers with strides greater than 1 reduces the dimensionality of the prediction vector, decreases the number of parameters, and improves computational efficiency. However, it also lowers temporal resolution, making it more difficult to detect short-duration earthquakes. Figure 16 illustrates this tradeoff by comparing the accuracy of models with output dimensions of (151×1), (76×1), (38×1), and (19×1). The figure presents two earthquake seismograms from the STEAD dataset, along with their corresponding time–frequency maps and the prediction vectors produced by the four models. The earthquake in Figure 16(a) has a source distance of 34.81 km and a duration of about 20 seconds. In this case, all models successfully detect the event, regardless of output resolution. In contrast, the earthquake in Figure 16(b) has a source distance of 2.05 km and a duration of only about 4 seconds. Here, models with lower output temporal

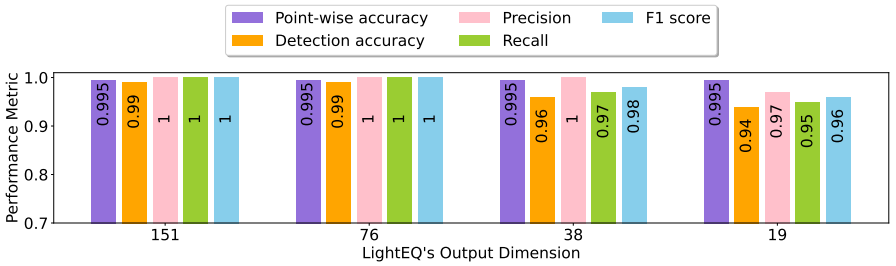


Fig. 17. Point-wise vs. detection accuracy: All four models have a 99.5% point-wise accuracy, but detection accuracy drops with the decrease in the output data points of the model.

resolution (e.g., 38x1 and 19×1) struggle to localize the short-duration signal accurately. Overall, this illustrates the tradeoff between the output resolution and the model's accuracy. Reducing the output dimension of the models results in fewer parameters but may incorrectly label small signals, thereby learning from mislabeled data. In Figure 17, we show that all models have a point-wise accuracy of 99.5%. However, the detection accuracy drops for models with output dimensions 38x1 and 19x1.

NN Architecture Search: Simply using neural architecture search (NAS) to select models with fewer parameters does not guarantee that they will be deployable on an MCU. The memory requirements depend on additional factors beyond parameter count. For example, in TensorFlow Lite for MCUs, we previously had to unroll RNNs into strictly feedforward structures for compatibility, which significantly increased memory consumption. The size of internal buffers also depends on the width of intermediate layers, and the impact of quantization on model accuracy is often unpredictable until it is actually performed. These implementation-specific details can only be resolved through experimentation and cannot be fully evaluated during NAS alone. As a result, selecting models for deployment involves trial and error to ensure they meet memory constraints in practice.

Detection over Phase Picking: In many scenarios, detecting potential seismic events directly on the sensor is sufficient to trigger further cloud-based analysis. While on-device phase picking, magnitude estimation, and source characterization are desirable, these tasks require dedicated training objectives, labels, and loss functions, and additional output heads to support higher-resolution analysis, leading to larger memory footprints. We constrain the model's peak RAM usage to below 100 kB to support deployment on low-cost, ultra-low-power MCUs. It is trained exclusively for event-level detection rather than precise P-wave onset estimation. We leave the integration of higher-resolution on-device tasks to future work.

Although more complex on-device models exist, such as SeismicSense (184 kB peak RAM), which performs phase picking in addition to detection, these approaches still do not achieve perfect accuracy and therefore cannot eliminate the need for cloud-based confirmation and analysis. Consequently, even advanced on-device phase detection methods require the transmission of event data for reliable validation, detailed processing, and long-term scientific use. Our hybrid approach balances efficiency and accuracy: lightweight models on the sensor filter out non-events and significantly reduce bandwidth utilization and power consumption, while the cloud performs compute-intensive tasks such as P- and S-phase classification, event localization, and magnitude estimation using larger and more accurate models.

STA/LTA as a Hard Pre-Filter: In our pipeline, STA/LTA acts as a hard pre-filter that determines whether a waveform segment is forwarded to the NN. Events that do not trigger STA/LTA are therefore not processed further. Section 5.3 evaluates different STA/LTA configurations and

analyzes the tradeoffs between precision, recall, and F1-score. Based on this evaluation, we select a high-sensitivity configuration that achieves 100% recall on the evaluated dataset and use it for all subsequent pipeline experiments. Under this configuration, STA/LTA primarily reduces computational load without missing earthquake events on STEAD dataset.

6 Conclusion

In this article, we present LightEQ, a resource-efficient, on-device earthquake detection pipeline that combines classical STA/LTA filtering with lightweight deep NNs, specifically designed for deployment on MCUs with limited computational and memory capacity. By addressing the challenges of operating in remote, bandwidth-constrained environments, LightEQ enables real-time seismic event detection on sensors without the need to transmit vast amounts of raw data to centralized servers. We design a set of lightweight models, of which the smallest operates within a 100 kB RAM budget, requiring 94 kB of RAM and 29k parameters and achieving up to 99% F_1 score. In comparison, SOTA models require significantly more resources to achieve similar performance. Our evaluation shows that it is not only feasible to deploy these lightweight models on resource-constrained embedded devices, but also cost-effective. The NN filters out most FPs from the STA/LTA detector, reducing the number of data transmissions, which are the most energy-intensive operation in the system. Occasional NN inference adds negligible energy overhead but contributes substantially to extending device lifetime. At higher sampling rates, frequent UART interrupts prevent the system from entering low-power states. Increasing the UART buffer size reduces interrupt frequency and enables batch processing; with a buffer of 100 measurements, battery life improves by 73%.

Acknowledgments

The authors also thank Bernhard Germann and Paul Wilke for their contributions to the implementation.

References

[1] Rex V. Allen. 1978. Automatic earthquake recognition and timing from single traces. *Bulletin of the Seismological Society of America* 68, 5 (1978).

[2] Ma. Cristina Arayata. 2024. Increased seismicity observed in Kanlaon Volcano. *Philippine News Agency* (July 2024). Retrieved from https://www.pna.gov.ph/articles/1228204

[3] TensorFlow Authors. 2024. TensorFlow Lite Micro. Retrieved March 11, 2026 from https://github.com/tensorflow/tflite-micro. Commit 9245002.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). Vol. 24, Curran Associates, Inc., 2546–2554. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[5] Ronald Newbold Bracewell and Ronald N. Bracewell. 1986. *The Fourier Transform and Its Applications*. McGraw-Hill New York.

[6] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. 2020. Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms. *IEEE Journal of Selected Topics in Signal Processing* 14, 4 (May 2020), 654–664. DOI: https://doi.org/10.1109/jstsp.2020.2969775

[7] Stuart Cheshire and Mary Baker. 2002. Consistent overhead byte stuffing. *IEEE/ACM Transactions on Networking* 7, 2 (2002), 159–172.

[8] Tai-Lin Chin, Chin-Ya Huang, Shan-Hsiang Shen, You-Cheng Tsai, Yu Hen Hu, and Yih-Min Wu. 2019. Learn to detect: Improving the accuracy of earthquake detection. *IEEE Transactions on Geoscience and Remote Sensing* 57, 11 (2019), 8867–8878. DOI: https://doi.org/10.1109/TGRS.2019.2923453

[9] Bernard A. Chouet. 1996. Long-period volcano seismicity: Its source and use in eruption forecasting. *Nature* 380, 6572 (1996).

[10] E. S. Cochran, J. Lawrence, A. E. Kaiser, B. Fry, A. I. Chung, and J. R. Evans. 2011. Comparison of the low-cost MEMS accelerometers used by the Quake-Catcher Network and traditional strong motion seismic sensors. In *Proceedings of the AGU Fall Meeting Abstracts*.

[11] Semtech Corporation. 2015. SX1276/77/78/79 Wireless, Sensing and Timing Datasheet. Retrieved from https://cdn-shop.adafruit.com/product-files/3179/sx1276_77_78_79.pdf. Accessed: 2025-08-18.
[12] Shuaiwen Cui, Yuguang Fu, Hao Fu, Xiao Yu, and Wei Shen. 2025. Smart adaptive trigger sensing powered by edge intelligence and digital twin for energy-efficient wireless structural health monitoring. *Mechanical Systems and Signal Processing* 241 (October 2025). DOI: https://doi.org/10.1016/j.ymssp.2025.113537
[13] Raquel S. Cabral, Andre L. L. Aquino, Alejandro C. Frery, Osvaldo A. Rosso, and Jaime A. Ramírez. 2013. Structural changes in data communication in wireless sensor networks. Retrieved from https://arxiv.org/abs/1308.3009
[14] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. 2021. Tensorflow lite micro: Embedded machine learning on tinyML systems. Retrieved from https://arxiv.org/abs/2010.08678
[15] Somnath Ghosh, Pramod Kumar Konugurthi, Gowri Shankar Rao Singupurapu, Shivi Patel, Tirupathi Tammanagari, Mallikarjuna Rao Desu, Lalit Krushna Thakar, and Ishika Ghara. 2021. On-board ship detection for medium resolution optical sensors. *Sensors* 21, 9 (2021). DOI: https://doi.org/10.3390/s21093062
[16] Gregory Giuliani, Hy Dao, Andrea de Bono, Bruno Chatenoux, Karin Allenbach, Pierric Laborie, Denisa Rodila, Nikos Alexandris, and Pascal Peduzzi. 2017. Live monitoring of earth surface (LiMES): A framework for monitoring environmental changes from earth observations. *Remote Sensing of Environment* 202, 1 (May 2017), 222–233. DOI: https://doi.org/10.1016/j.rse.2017.05.040
[17] Global Volcanism Program. 2019. Report on Ubinas (Peru) – September 2019. *Bulletin of the Global Volcanism Network* (10 September 2019). Retrieved March 11, 2026 from https://volcano.si.edu/showreport.cfm?doi=10.5479/si.GVP.BGVN201909-354020
[18] D. Griffin and Jae Lim. 1984. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32, 2 (1984), 236–243. DOI: https://doi.org/10.1109/TASSP.1984.1164317
[19] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360*. Retrieved from https://arxiv.org/abs/1602.07360
[20] Marius P. Isken, Jens Karstens, Paraskevi Nomikou, Michelle Maree Parks, Vincent Drouin, Eleonora Rivalta, Gareth J. Crutchley, Mahmud Haghshenas Haghighi, Emilie E. E. Hooft, Simone Cesca, et al. 2025. Volcanic crisis reveals coupled magma system at Santorini and Kolumbo. *Nature* 645, 8082 (2025), 939–945. DOI: 10.1038/s41586-025-09525-7
[21] Petar Jakuš and Hrvoje Džapo. 2025. Implementing keyword spotting on the MCUX947 microcontroller with integrated NPU. *arXiv preprint arXiv:2506.08911*. Retrieved from https://arxiv.org/abs/2506.08911
[22] Supriya Kapur, Asit Mishra, and Debbie Marr. 2017. Low precision RNNs: Quantizing RNNs without losing accuracy. *arXiv preprint arXiv:1710.07706*. Retrieved from https://arxiv.org/abs/1710.07706
[23] Lion Krischer, Tobias Megies, Moritz Beyreuther, Robert Barsch, Joachim Wassermann, et al. 2024. ObsPy: STA/LTA triggering documentation. Retrieved March 12, 2026 from https://docs.obspy.org/packages/autogen/obspy.signal.trigger.classic_sta_lta.html
[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Vol. 25, Curran Associates, Inc., 1097–1105. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[25] Liangzhen Lai, Naveen Suda, and Vikas Chandra. 2018. CMSIS-NN: Efficient neural network kernels for arm Cortex-M CPUs. *arXiv preprint arXiv:1801.06601*. Retrieved from https://arxiv.org/abs/1801.06601
[26] Thorne Lay and Terry C. Wallace. 1995. *Modern Global Seismology*. Elsevier.
[27] Gregor Lenz and Douglas McLelland. 2024. Low-power ship detection in satellite images using neuromorphic hardware. *arXiv preprint arXiv:2406.11319*. Retrieved from https://arxiv.org/abs/2406.11319
[28] Jongseong Lim, Sunghun Jung, Chan JeGal, Gwanghoon Jung, Jung Ho Yoo, Jin Kyu Gahm, and Giltae Song. 2022. LEQNet: Light earthquake deep neural network for earthquake detection and phase picking. *Frontiers in Earth Science* 10, 1 (2022), 848237. DOI: https://doi.org/10.3389/feart.2022.848237
[29] Guojin Liu, Rui Tan, Ruogu Zhou, Guoliang Xing, Wen-Zhan Song, and Jonathan M. Lees. 2013. Volcanic earthquake timing using wireless sensor networks. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*.
[30] Han Liu and Jian-zhong Zhang. 2014. STA/LTA algorithm analysis and improvement of microseismic signal automatic detection. *Progress in Geophysics* 29, 4 (2014), 1708–1714.
[31] Tianlin Liu, Jannes Münchmeyer, Laura Laurenti, Chris Marone, Maarten V. de Hoop, and Ivan Dokmanić. 2024. SeisLM: A foundation model for seismic waveforms. *arXiv preprint arXiv:2410.15765*. Retrieved from https://arxiv.org/abs/2410.15765
[32] Thomas Lorenser. 2016. The DSP capabilities of arm Cortex-M4 and cortex-M7 processors. 1–19. Retrieved from https://developer.arm.com/documentation/102812/latest

[33] James Luetgert and David Oppenheimer. 2010. The NetQuakes project–research–quality seismic data transmitted via the internet from citizen-hosted instruments. *AGU Fall Meeting Abstracts* 2010 (December 2010), S51E–03.

[34] S. Mostafa Mousavi, William L. Ellsworth, Weiqiang Zhu, Lindsay Y. Chuang, and Gregory C. Beroza. 2020. Earthquake transformer—An attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature Communications* 11, 1 (2020), 3952.

[35] S. Mostafa Mousavi, Yixiao Sheng, Weiqiang Zhu, and Gregory C. Beroza. 2019. Stanford earthquake dataset (STEAD): A global data set of seismic signals for AI. *IEEE Access* 7 (2019), 179464–179476. DOI: https://doi.org/10.1109/ACCESS.2019.2947848

[36] S. Mostafa Mousavi, Yixiao Sheng, Weiqiang Zhu, and Gregory C. Beroza. 2019. STanford EArthquake Dataset (STEAD) Repository. Retrieved from https://github.com/smousavi05/STEAD. Accessed: 2026-02-15.

[37] S. Mostafa Mousavi, Weiqiang Zhu, Yixiao Sheng, and Gregory C. Beroza. 2019. CRED: A deep residual network of convolutional and recurrent units for earthquake signal detection. *Scientific Reports* 9, 1 (2019), 10267. Retrieved from https://arxiv.org/abs/1810.01965

[38] Nordic Semiconductor. 2024. *nRF52840 Product Specification*. Retrieved from https://docs.nordicsemi.com/bundle/nRF52840_PS_v1.9/resource/nRF52840_PS_v1.9.pdf. Accessed: 2025-02-19.

[39] Nordic Semiconductor. 2024. *nRF5340 Product Specification*. Retrieved from https://docs.nordicsemi.com/bundle/ps_nrf5340/page/keyfeatures_html5.html. Accessed: 2025-02-19.

[40] Thomas E. Norris, Garu L. Hart, Eric H. Larson, Peter Tarczy-Hornoch, David L. Masuda, Sherrilynne S. Fuller, Peter J. House, and Sarah M. Dyck. 2002. Low-bandwidth, low-cost telemedicine consultations in rural family practice. *The Journal of the American Board of Family Practice* 15, 2 (2002), 123–7. Retrieved from https://api.semanticscholar.org/CorpusID:6098244

[41] Thibaut Perol, Michaël Gharbi, and Marine Denolle. 2018. Convolutional neural network for earthquake detection and location. *Science Advances* 4, 2 (2018), e1700578.

[42] Adam T. Ringler and Patrick Bastien. 2020. A brief introduction to seismic instrumentation: Where does my data come from? *Seismological Research Letters* 91, 2A (2020), 1074–1083.

[43] Omar M. Saad and Yangkang Chen. 2020. Earthquake detection and P-wave arrival time picking using capsule neural network. *IEEE Transactions on Geoscience and Remote Sensing* 59, 7 (2020), 6234–6243.

[44] Sandra Sendra, Jaime Lloret, Jose Miguel Jimenez, and Lorena Parra. 2015. Underwater acoustic modems. *IEEE Sensors Journal* 16, 11 (2015), 4063–4071.

[45] Jeonguk Seo, Yunu Kim, Jisung Ha, Dongyoup Kwak, Minsam Ko, and Mintaek Yoo. 2024. Unsupervised anomaly detection for earthquake detection on Korea high-speed trains using autoencoder-based deep learning models. *Scientific Reports* 14, 1 (2024), 639.

[46] B. K. Sharma, Amod Kumar, and V. M. Murthy. 2010. Evaluation of seismic events detection algorithms. *Journal of the Geological Society of India* 75, 3 (2010), 533–538.

[47] Peter M. Shearer. 2019. *Introduction to Seismology*. Cambridge University Press.

[48] Kuan-Wei Tang and Kuan-Yu Chen. 2026. SeismoDual: A dual-domain deep learning framework for robust seismic phase picking. *Computers & Geosciences* 207 (2026), 106080. DOI: https://doi.org/10.1016/j.cageo.2025.106080

[49] Hasan Tariq, Farid Touati, Mohammed Abdulla E. Al-Hitmi, Damiano Crescini, and Adel Ben Mnaouer. 2019. A real-time early warning seismic event detection algorithm using smart geo-spatial bi-axial inclinometer nodes for Industry 4.0 applications. *Applied Sciences* 9, 18 (2019), 3650.

[50] Tenergy. 2023. 14.8V 13.2Ah Lithium Ion Battery Pack. Retrieved from https://www.tenergy.com/14-8v-13200mah-li-ion-battery-pack. Accessed: 2024-12-01.

[51] Google TensorFlow. 2016. Large-scale machine learning on heterogeneous systems. Retrieved from https://arxiv.org/abs/1603.04467

[52] Tomoki Tokuda and Hiromichi Nagao. 2023. Seismic-phase detection using multiple deep learning models for global and local representations of waveforms. *Geophysical Journal International* 235, 2 (2023), 1163–1182.

[53] Amadej Trnkoczy. 2009. Understanding and parameter setting of STA/LTA trigger algorithm. In *Proceedings of the New Manual of Seismological Observatory Practice (NMSOP)*.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. Retrieved from https://arxiv.org/abs/1706.03762

[55] Peter H. Voss and Lars Ottemöller. 2015. Seismic network and data quality. In *Encyclopedia of Earthquake Engineering*, Michael Beer, Ioannis A. Kougioumtzoglou, Edoardo Patelli, and Siu-Kui Au (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2920–2932. DOI: https://doi.org/10.1007/978-3-642-35344-4_193

[56] Jian Wang, Zhuowei Xiao, Chang Liu, Dapeng Zhao, and Zhenxing Yao. 2019. Deep learning for picking seismic arrival times. *Journal of Geophysical Research: Solid Earth* 124, 7 (2019), 6612–6624. DOI: https://doi.org/10.1029/2019JB017536

[57] Yue Wu, Youzuo Lin, Zheng Zhou, David Chas Bolton, Ji Liu, and Paul Johnson. 2019. DeepDetect: A cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing* 57, 1 (2019), 62–75. DOI: https://doi.org/10.1109/TGRS.2018.2852302

[58] Clara E. Yoon, Ossian O'Reilly, Karianne J. Bergen, and Gregory C. Beroza. 2015. Earthquake detection through computationally efficient similarity search. *Science Advances* 1, 11 (2015), e1501057. DOI: https://doi.org/10.1126/sciadv.1501057

[59] Xiao Yu, Shuaiwen Cui, Yuguang Fu, and Qi Zhang. 2026. Decentralized system-centric sensor fault diagnosis and recovery using edge computing for wireless structural health monitoring systems. *Measurement* 261 (2026), 119994. DOI: https://doi.org/10.1016/j.measurement.2025.119994

[60] Tayyaba Zainab, Laura Harms, Jens Karstens, and Olaf Landsiedel. 2025. SeismicSense: Phase picking of seismic events with embedded machine learning. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*.

[61] Tayyaba Zainab, Jens Karstens, and Olaf Landsiedel. 2023. LightEQ: On-device earthquake detection with embedded machine learning. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*.

[62] Zephyr Project. 2024. Zephyr RTOS. Retrieved from https://github.com/zephyrproject-rtos/zephyr. Accessed: 2025-02-19.

[63] Yu Zhao, Pan Deng, Junting Liu, Mulan Wang, and Jiafu Wan. 2022. LCANet: Lightweight context-aware attention networks for earthquake detection and phase-picking on IoT edge devices. *IEEE Systems Journal* 16, 3 (2022), 3558–3569. DOI: https://doi.org/10.1109/JSYST.2021.3075637

[64] Weiqiang Zhu and Gregory C. Beroza. 2019. PhaseNet: A deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International* 216, 1 (2019), 261–273.

Received 8 September 2025; revised 30 January 2026; accepted 12 February 2026