RESEARCH-ARTICLE

LightEQ: On-Device Earthquake Detection with Embedded Machine Learning

TAYYABA ZAINAB, University of Kiel, Kiel, Schleswig-Holstein, Germany

JENS KARSTENS, GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, Schleswig-Holstein, Germany

OLAF LANDSIEDEL, University of Kiel, Kiel, Schleswig-Holstein, Germany

Open Access Support provided by:

University of Kiel

GEOMAR Helmholtz Centre for Ocean Research Kiel

Published: 09 May 2023

Citation in BibTeX format

IoTDI '23: International Conference on Internet-of-Things Design and Implementation
May 9 - 12, 2023
TX, San Antonio, USA

Conference Sponsors:
SIGBED

https://doi.org/10.1145/3576842.3582387
ISBN: 9798400700378

LightEQ: On-Device Earthquake Detection with Embedded Machine Learning

Tayyaba Zainab
tzainab@geomar.de
tza@informatik.uni-kiel.de
GEOMAR Helmholtz Centre for Ocean Research Kiel
& University of Kiel
Germany

Jens Karstens
jkarstens@geomar.de
GEOMAR Helmholtz Centre for Ocean Research Kiel
Germany

Olaf Landsiedel
ol@informatik.uni-kiel.de
University of Kiel
Germany

ABSTRACT

The detection of earthquakes in seismological time series is central to observational seismology. Generally, seismic sensors passively record data and transmit it to the cloud or edge for integration, storage, and processing. However, transmitting raw data through the network is not an option for sensors deployed in harsh environments like underwater, underground, or in rural areas with limited connectivity. This paper introduces an efficient data processing pipeline and a set of lightweight deep-learning models for seismic event detection deployable on tiny devices such as microcontrollers. We conduct an extensive hyperparameter search and devise three lightweight models. We evaluate our models using the Stanford Earthquake Dataset and compare them with a basic STA/LTA detection algorithm and the state-of-the-art machine learning models, i.e., CRED, EQtransformer, and LCANet. For example, our smallest model consumes 193 kB of RAM and has an F1 score of 0.99 with just 29k parameters. Compared to CRED, which has an F1 score of 0.98 and 293k parameters, we reduce the number of parameters by a factor of 10. Deployed on Cortex M4 microcontrollers, the smallest version of LightEQ-NN has an inference time of 932 ms for 1 minute of raw data, an energy consumption of 5.86 mJ, and a flash requirement of 593 kB. Our results show that resource-efficient, on-device machine learning for seismological time series data is feasible and enables new approaches to seismic monitoring and early warning applications.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; **Client-server architectures**; • **Computing methodologies → Neural networks**.

KEYWORDS

Seismological data analysis, Earthquake detection, On-device, Deep Neural Networks, Low-Power, Internet of Things, Edge AI

IoTDI '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0037-8/23/05…$15.00
https://doi.org/10.1145/3576842.3582387

ACM Reference Format:
Tayyaba Zainab, Jens Karstens, and Olaf Landsiedel. 2023. LightEQ: On-Device Earthquake Detection with Embedded Machine Learning. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3576842.3582387

1 INTRODUCTION

A dense, global network of seismic sensors monitors the earth's surface and helps to understand natural phenomena like earthquakes, volcanic eruptions, and landslides, as well as subsurface operations such as oil and gas drilling, the storage of CO_2, and hydro-thermal energy production [7]. Typically, these sensors send their raw data continuously to a central facility with sufficient computing power, which then conducts analyses to constrain geophysical properties or to operate early warning systems.

In the case of sensor deployments in underwater, sub-terrain, or rural environments, such centralized data analysis is not possible due to constrained connectivity. For example, conventional seismic sensors deployed submarine, so called Ocean-bottom seismometers [2], merely record raw data, which will be analyzed only after recovery of the entire instrument. The limited communication range (0.1 - 6 km), bandwidth of a few kbit/s and high-energy demands of acoustic underwater communication hinders a continuous stream of raw data to the cloud [30], while electromagnetic data transfer is not possible under water. Similarly, deployments in rural areas often cannot provide the required communication bandwidth, as cellular coverage in such areas is commonly limited and satellite internet solutions are costly and energy intensive [13]. For such deployments, seismic sensors have to be collected manually to analyze their recordings.

To enable deployment of seismic sensors in such harsh settings without the need for costly recovery, this paper introduces LightEQ, a data pre-processing pipeline and a set of lightweight deep-learning models deployable on tiny devices such as microcontrollers, see Figure 1. LightEQ enables on-device analysis of the event stream and detection of seismic events, i.e., earthquakes. LightEQ filters relevant events from a continuous data stream and, in case further analyses in the cloud is required, allows minimizing the required data transfer with low-bandwidth underwater, cellular, or satellite communication.

Utilizing neural network architecture search and extensive hyperparameter optimization, we devise multiple lightweight models. These are tailored to resource-constrained nodes in the Internet of

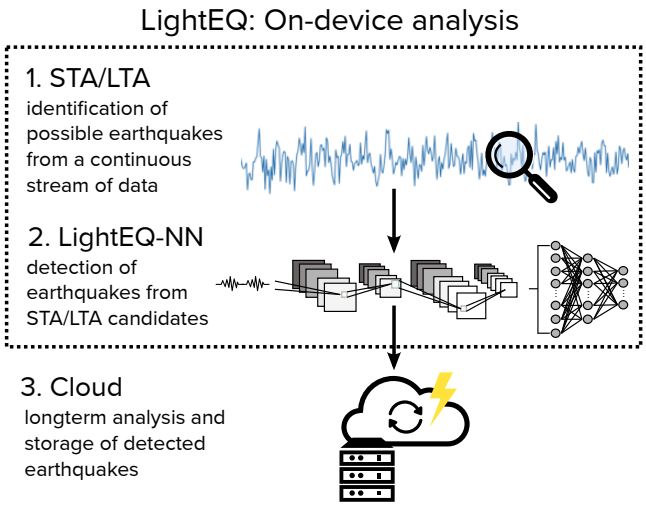

Figure 1: Overview of LightEQ. STA/LTA performs on-device identification of potential earthquakes and operates to a continuous stream of seismic data. Once a frame is identified as a potential earthquake, it is forwarded to a lightweight neural network to perform on-device classification; After classification, the event and associated metadata may be sent to the cloud over a low-bandwidth connection for long-term storage and analysis.

Things (IoT) and hit a sweet spot by achieving both high accuracy and minimal resource requirements in terms of computation time and memory. After quantization and compression, our models become sufficiently small to be deployable on small microcontrollers, such as Cortex M4, with less than 256 kB of RAM. With a configurable parameter count of 29k and 73k, we achieve an accuracy of 0.98 and 0.95, respectively. Larger Cortex M4 models, i.e., with upto 512 kB of RAM, can also deploy a model with 151k parameters with 0.98 accuracy. On the microcontroller (MCU), we achieve a minimum inference time of 0.932s and energy consumption of 5.86 mJ for 1 minute of raw data, utilizing the built-in Digital Signal Processor (DSP) for NN inference. Overall, LightEQ-NN achieves an on-par accuracy compared to state-of-the-art models, such as CRED [26], EQtransformer [22], and LCAnet [37], while requiring significantly fewer parameters, computation power and energy consumption, which are key requirements for the on-device NN detection.

Further, a key challenge in seismic event detection is that events are rare, and thereby a continuous operation of any neural network for on-device event detection is inefficient. Instead, we introduce a two-stage pipeline as system architecture in LightEQ: First, we filter the continuous data stream of our seismic sensors for possible seismic events using short-time-average against long-time-average filtering (STA/LTA). STA/LTA is a lightweight band pass filter and allows for resource-efficient identification of potential earthquake events [20]. We then forward potential earthquake events to our LightEQ-NN model for separation of noise from actual seismic events, i.e., the detection of earthquakes. We release LightEQ as open-source to the public[1].

Contributions. This paper makes the following contributions:

(1) We show that it is possible to perform on-device earthquake classification of seismic data on resource-constrained IoT devices.
(2) We introduce a data processing pipeline consisting of STA/LTA pre-filtering for identification of possible earthquakes and a lightweight NN based event detection.
(3) We present LightEQ-NN, a lightweight earthquake detection model which is small enough to be deployed on a microcontroller with 512 kB of RAM and can detect earthquake events from seismic data.
(4) We show that our quantized classifier can detect seismic earthquake events for STEAD dataset with 98% accuracy on a low-power platform and uses flash memory up to 726 kB for model with 151k parameters.
(5) Our integer quantized models with DSP accelerator enabled can reduce the inference time by a factor of 6.7 compared to DSP accelerator disabled and can be deployed even on basic computation units as it has 0 floating-point operations.

Outline. We provide the required background in §2 and summarize the related literature in §3. We present the design and the selected models in §4 and in §5 we conduct an in-depth evaluation of LightEQ-NN. §6 concludes the paper.

2 BACKGROUND

In the following, we briefly introduce the required background on earthquake detection and embedded neural networks.

2.1 Earthquakes

Earthquakes are caused by the sudden release of stress and seismic energy accumulated between tectonic plates of the earth as the result of their relative motion. The seismic energy radiates in the form of seismic waves, which can be detected by seismic sensors, e.g., seismometers, and are commonly recorded along three axes, defining 3-component seismological data: (1) the Z-component records the vertical motion, the (2) N-component records motion in north/south direction, and (3) the E-component, records the motion in east/west direction.

2.2 Frequency Spectrogram

Fourier transform computes the frequency spectrogram of a given input time series, which often helps to characterize signals. For example, earthquakes often have a characteristic frequency content when compared to background noise [3]. Short-time Fourier transform (STFT), in turn, computes a time-localized frequency spectrogram, i.e, the frequency representation for a time window. Thus, STFT segments the signal into narrow time intervals and computes Fourier transforms for each segment and integrates the amplitude spectra for each time step into a 2D matrix of frequency versus time, referred to as a spectrogram or frequency-time map [8]. While the general FT only allows recognizing the occurrence of an event, the STFT allows determining its timing, which is required for our application.

[1]Here is the link to the GitHub repository: https://github.com/ds-kiel/LightEQ.git.

2.3 Neural Networks

LightEQ employs Convolutional Neural Networks (CNNs) [27] and Recurrent Neural Networks (RNNs) [32], both are established approaches for analysis of time series data, such as seismic measurements. Convolutional layers consist of a convolution kernel, filter, and stride. A convolution kernel is a matrix that applies a linear transformation to an image and strides to down-sample the matrix to lower dimensions. RNN layers such as a Long short-term memory (LSTM) can remember information for extended periods; hence it is suitable for long-time series data [15].

2.4 Quantization

A quantized neural network [5] converts the weights from high-precision floating points to integers. The mapping function reduces the number of bits used to represent the weights and activations. This can be applied during training of the model as well as post training. However, quantization may have a negative effect on the accuracy of NN applications [11]. To date, RNN are not fully supported by training aware quantization in TensorFlow Lite for MCUs, which we are using as a library in our implementation. Also, when quantizing a model with RNNs, mapping errors can accumulate and result in a variable accuracy degradation[12].

3 RELATED WORK

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

3.1 Model-Driven Analysis

There are several approaches to event detection in seismological signal analysis. One basic technique is "threshold triggering," which detects events based on a user-defined threshold limit. However, this type of technique is highly sensitive to noise and does not work in many environments [31, 36]. A more robust approach is the "short-time average/long-time average" (STA/LTA) method [33, 36]. STA/LTA compares the average energy in a short-term average leading window to a long-term average trailing window and calculates the ratio of STA to LTA. If the calculated ratio is greater than a certain threshold, a detection is triggered. Compared to the threshold triggering, STA/LTA allows event detection even under elevated noise levels. STA/LTA performs well with impulsive, high signal-to-noise (SNR) signals but it will be inefficient under conditions with low SNR or elevated cultural noise, which may trigger false positives. In this paper, we use STA/LTA as a pre-filter, as it causes many false positives on noisy data but also practically zero false negatives, when configured sufficiently sensitive. Another approach is "template matching" [36], also known as filter matching, which exploits the 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 "fingerprinting and similarity threshold" (FAST) [36] use "locality sensitive hashing" (LSH) to improve the efficiency of the similarity search. Lately, these traditional methods have been replaced by data-driven approaches and most recently by neural networks.

3.2 Data-Driven Analysis

In recent years, machine learning has become an integral part of seismological data analysis [4, 22, 26]. Chin et al. (2019) [4] utilize three machine learning algorithms for earthquake detection: K-nearest neighbor, classification tree, and support vector machine (SVM). K-nearest neighbor is the simplest earthquake classifier and calculates the Euclidean distance between the new observation and the feature record. This classification is driven by a majority votes of Top K shortest distance records. The classification tree builds a tree using the training data, where each branch of the tree is a feature cut-point, which leads 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 that modern approaches based on neural networks achieve.

3.2.1 Neural Networks for the Cloud. Recent works [28, 29, 35] propose the usage of deep CNNs, while others (e.g. CRED [26]) suggest combining CNNs with RNNs. In CRED, features are extracted at the CNN layers and sequence learning is done by uni and bi-directional LSTMs. The reported training and validation accuracy of CRED is 99.33% and 99.24%, respectively. Other neural network-based earthquake detectors like EQTransformer [22], on the other hand, apply an attention mechanism and achieve an even higher accuracy of up to 100%. Nonetheless, all approaches are quite resource-demanding and not suitable for deployment on resource-constrained devices, as we show in our evaluation in §5.

3.2.2 Neural Networks for the Edge. LCANet [37] is a deep learning model inspired by SqueezeNet [10]: An image classifier that reduces the number of parameters by 50% by replacing the 3x3 convolution filter with 1x1 and still has a similar performance as the original, non-reduced model (AlexNet [17]). whereas, Google proposed MobileNets [9] using an extensive hyperparameter search. LCANet [37] uses context-aware attention modules and optimizes the deep learning model to reduce the computational requirement to be able to fit onto an edge device.

Seismology has also adopted smartphones to detect earthquakes. MyShake [16] uses volunteers' cellphones as a seismic station/ earthquake detector, processes the data using a neural network, and sends it to the server. The limitation to Myshakes is that it heavily relies on the volunteer's usage of the cellphone and also requires smartphones that can run NN with vast amounts of energy supply. Prior to MyShake, NetQuakes [21] was another project for detecting earthquakes that uses MEMS sensors.

3.2.3 Neural Networks for the Sensor Plane. LEQNet [19] proposes a deep neural network model that uses a deeper bottleneck, recursive structure and depthwise separable CNN which reduces the model size by 87.68% as compared to EQTransformer [22] and have the F1 score of 0.99. However, their model is based on a transformer architecture, which is not available in today's frameworks for embedded devices such as Tensorflow-Lite, due to its complexity and heavyweight nature is for MCUs [6]. Further, even though their model reduces the number of parameters, we show in our design and evaluation, that the number of parameters alone is not the

decisive measure for deploying models on a resource-constrained device. In contrast, our analyses reveal that the types of layers, such as recurrent or convolutional, their size and intermediate data structures, have a strong impact on the resource demands.

Overall, we argue that present-day models for earthquake detection are too heavyweight for resource-constrained devices and closing this gap by introducing a set of lightweight models with detection accuracy of 98% to 99% tailored for MCUs.

4 DESIGN: LIGHTEQ

With LightEQ, we perform an on-device classification of seismic data into earthquake or noise. In the following, we first discuss design challenges in §4.1. Next, we give an overview of LightEQ in §4.2, and detail on our data possessing pipeline in §4.3. In §4.4, we discuss the basic neural network architecture of LightEQ-NN. Next, we employ neural network architecture search in §4.5 to derive specific configurations, which we then introduce in §4.6. We discuss quantization in §4.7 and implementation in §4.8.

4.1 Design Challenges

When working with resource-constrained devices, we need to carefully evaluate the number of layers and their configurations, as each additional parameter increases the memory and computations demands of the neural network while also promises a higher detection accuracy. Furthermore, as we work with time series data, inference time is vital when selecting a model; a bigger model in terms of model size, output dimension, and NN layers typically also results in a larger inference time. RNN contributes to most of the model's parameters compared to the convolutional layers. Considering the flash consumption, choosing a model for the microcontroller is not straightforward. Number of parameters or model_size is proportional to flash memory consumption, bigger models require more flash memory. However, RAM is different and often needs to be determined by experimentation. Unrolling of LSTMs is common on embedded hardware, which further increases flash requirements and makes the RAM memory consumption hard to predict. We employ extensive neural network architecture search and hyperparameter search to identify models that meet these demands.

Quantizing the model, i.e., its weights, to an integer representation, for example from 32-bit float to bit-8 integer, as common for an embedded device, reduces the size of the model by a factor of four. Further, it allows the usage of accelerators for integer matrix multiplications, which are becoming increasingly available on modern microcontrollers. However, the usage of quantization has to be done carefully, as it can result in accuracy degradation. Moreover, this reduction is variable in our case (2% - 20%), as precision errors can accumulate in the RNN layer [14]. Finally, inference with neural networks is energy intensive. We need to devise an efficient system architecture where we pre-filter the data stream of our seismic sensors for possible earthquakes, which we then classify with our neural network.

4.2 System Overview

The on-device data analysis in LightEQ consists of six main steps, see Figure 2: Filtering (STA/LTA) and pre-processing (STFT) followed by neural network-based feature extraction, sequence learning and classification and as final step threshold filtering. The seismic sensor records the ground motions for the N, E, and Z-components and continuously performs STA/LTA filterting. Once the STA/LTA triggers and identifies a potential earthquake event, the corresponding time series is passed to the pre-processing unit, where each of the three raw seismic time series is transformed from the time domain into frequency spectrograms using STFT. Next, we forward this frequency spectrogram into the LightEQ-NN model for classification. The output of our model is a vector of predictions for a time window, from which we then compute an integrated binary classification using a threshold filter.

4.3 Signal Pre-Processing

Before the actual neural network, we have two signal pre-processing steps: the detection of earthquake candidates via STA/LTA, followed by a conversion to the frequency domain.

First, we employ the established STA/LTA method as a filter to identify possible earthquakes. This is driven by the following two observations: First, earthquake are very rare events and therefore only account for a small fraction of the actual measurements. Second, as we show in our evaluation, even the small neural networks devised in this paper takes a couple of seconds to execute for each minute of raw data. Thus, due to the sparsity of the events and computational cost, we employ the lightweight filtering of STA/LTA as a method to identify possible earthquake in the sensor data. We have shown in our evaluation that we can achieve 100% recall and 88% precision in this first step with an overly sensitive configured filter. While we acknowledge that STA/LTA is an established approach for earthquake detection, the novelty here is that we use it as a pre-filter in our system architecture to identify potential earthquakes before conducting actual detection using a neural network.

After a possible earthquake has been detected, we transform the corresponding time window to the frequency domain using short-term Fourier transformation (STFT) for further analysis. Inputs to the system are the three-component seismic data with a sampling frequency of 100 Hz and a segment length of 60 seconds. This three-dimensional time series data is converted to a spectrogram using STFT with a sampling frequency of 100 Hz and a length of each segment of 80, resulting in matrices with dimensions of (151x41x3). The spectrogram provides time-frequency distributions of the seismic energy, which are indicative of seismic signals from specific seismic sources. Studies [23, 24] show that time-frequency transformation results in better characterizing seismic signals and we use this signal as input to our neural network in the next stage.

4.4 Basic Neural Network Architecture

The NN architecture of LightEQ-NN combines (1) a feature extraction block, which extracts characteristic features from the data, (2) a time series sequence learning block, (3) and a classification block that calculates an earthquake probability vector for sub-frames of the input, see Figure 3. Finally, a threshold filtering function computes the number of earthquake signal detections in the seismic time series and outputs a binary decision (earthquake or no earthquake). In the following, we will discuss the overall architecture, the exact number of layers and their configurations that are derived from neural network architecture search at a later stage. The overall architecture is inspired by CRED, but the NN framework

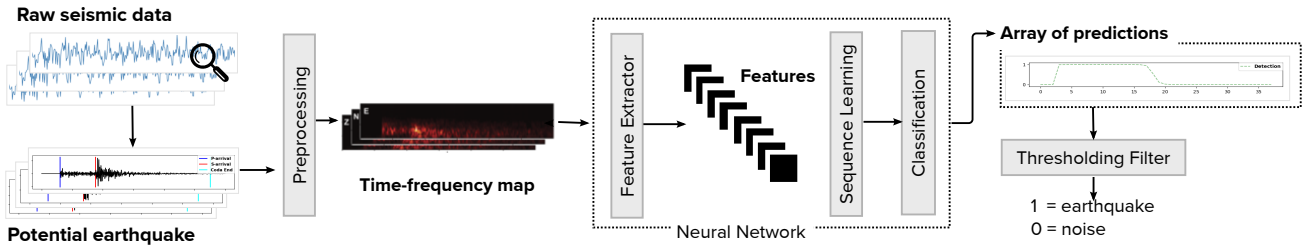

Figure 2: System design of LightEQ. Once a seismic signal is classified as a potential earthquake by STA/LTA, it passes to a pre-processing unit, which converts it into a time-frequency map. Next, we classify the data with a neural network. The output of the neural network is a prediction vector for sub frames. Finally, a threshold filter classifies the overall time frame as noise or earthquake.

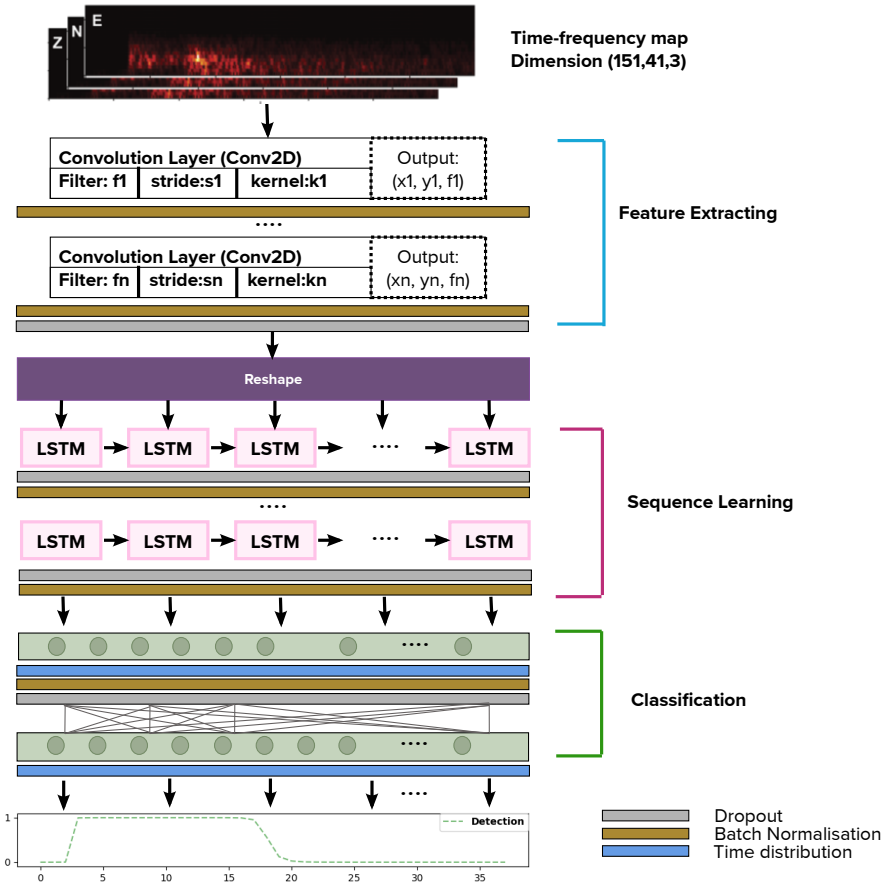

Figure 3: The basic neural network architecture of LightEQ-NN consists of three components. (1) CNN layers for feature extraction, regular or residual blocks depending on configuration; (2) LSTMs for sequence learning; and (3) classification layers.

significantly extended beyond it to match the demanding resource constraints of embedded devices.

Feature extraction: We combine the three spectrograms for each component of the seismic data into one matrix (151x41x3) and transfer it to the neural network. The resulting three-component spectrogram is then passed to through the feature extraction sub-block, which consists of convolution layers with a rectified linear unit (ReLU) as an activation function. Each convolutional layer is followed by a batch normalization layer that normalizes the output of the convolutional layer. The final layer is a dropout layer,

Table 1: Hyperparameter search space we use for LightEQ.

Design Dimension		Range	# of Choices
Filter size		8, 16, 32	3
Kernel Size		3, 5, 7, 9	4
Stride Size		1, 2	2
Selection	# of Conv. layers	1, 2, 3, 4, 5	5
	# of Conv. blocks	3, 4, 5	3
# 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

which stabilizes the learning process, prevents over-fitting, and significantly reduce training time [18].

Sequence learning: The feature extraction is followed by unidirectional LSTM layers, which learn sequential patterns with the help of their internal memory for sequence-to-sequence data. Each LSTM layer is followed by a dropout layer to avoid vanishing gradients, which hampers the learning of long data sequences, and batch normalization.

Classification: Finally, the last two layers of our model are the classification layers. The first one uses ReLU activation function and the last layer uses the sigmoid activation function. These two fully connected layers perform high-level reasoning and map the learned sequence model to output a vector of predicted probabilities for each sample containing an earthquake signal, see Figure 5.

Threshold Filtering: The output of LightEQ-NN is a prediction vector for each sample of the input time series. To arrive at a binary decision, we apply threshold filtering. If there is at least one event detected in the input data, it is classified as an earthquake.

4.5 Neural Network Architecture Search

Building on the basic network architecture introduced in the previous section, we next conduct an extensive neural network architecture and hyperparameter search of 400 trials to identify a set of models with high accuracy and minimal loss. However, the design space for this architecture is vast and consists of many possible configurations, as shown in Table 1. Therefore, performing a brute-force search for the best possible parameters is infeasible. Alternatively, we adopt Bayesian optimization, i.e., Tree-structured Parzen Estimator (TPE) [1] search, which approximates the performance of hyperparameters based on historical measurements and subsequently chooses new hyperparameters for further testing.

We perform two searches:

1) By varying the number of CNN blocks for feature extraction, along with choosing the filter size of 8, 16, or 32, kernel size of 3, 5, 7, or 9, as well as the stride of either 1 or 2 for each block individually. Each CNN block has three CNN layers, where the first layer has the filter size, kernel size, and stride suggested by the parameter search. The other two CNN layers have the same filter size, but the kernel is reduced to 2, and the stride is fixed to 1 to keep the output dimension consistent; 2) By varying the number of CNN layers. This search does not use the residual blocks. Instead, the hyper-parameter search suggests filter size, kernel size, and strides for all the layers, using the same filter, kernel, and stride options as described for 1).

A dropout layer follows these CNN blocks/layers with a dropout rate chosen from a uniform distribution range of 0.3 to 0.7. For the LSTM layer/s for sequence learning, we have two options: 32 and 64 units in our search. Other layers, such as dropout layers, can have a dropout rate from a uniform distribution of 0.5 to 0.9, followed by the classification block, which remains the same throughout the search. Hyperparameters, such as batch size, can be configured as 50, 80, and 100, and epochs can be set as 20, 30, or 40. Depending on the number of CNN blocks/layers and the stride of the model, the output of our models is a probability vector of dimensions 19x1, 38x1, 76x1, or 151x1.

4.6 Selected Neural Network Models

We conduct an extensive neural network architecture search and hyperparameter optimization, resulting in the derivation of three models, see Figure 4. During the search, we consider test accuracy, the number of parameters, and output dimensions as key metrics and limit our choices based on RAM and Flash consumption. The three selected models currently have a prediction vector accuracy of 99.5%:

- Model_(1,1) has a detection accuracy of 98.5%, 29k total parameters, and an output dimension of 76.
- Model_(5Res,1) has a detection accuracy of 96%, 73k total parameters and an output dimension of 36.
- Model_(3Res,1) has a detection accuracy of 99.5%, 151k total parameters and the output dimension of 76.

Model_(1,1) This model has one CNN layer with a kernel of 7x7, a filter of 8, and a stride of 2, which changes the output dimension of the model from 151 to 75, which is followed by batch normalization, a Relu activation layer, and a dropout layer with a dropout rate of 0.31. Next is a single LSTM layer with 32 units and a dropout layer with a rate of 0.61. Finally, the model has two classification layers, i.e., a dense layer with 1 unit following a dense layer with 64 units and a drop out layer in between with a rate of 0.89.

Model_(5Res,1) This model has five CNN residual blocks and one LSTM layer. Each CNN block consists of three CNN layers, see Figure 4 for details about kernel size, stride and filter length. These blocks are followed by a dropout layer with a rate of 0.59 and an LSTM layer with 32 units. Next we have a dropout layer with a rate of 0.59. Lastly, the model has a dense layer with 64 units with a dropout rate of 0.58 and a dense layer with 1 unit.

Model_(3Res,1) This model has three CNN residual blocks and one LSTM layer. Each CNN block also consists of three CNN layers, see Figure 4 and a dropout layer with a rate of 0.4. This is followed by a LSTM layer with 32 units and a dropout layer with a rate of 0.73. The model has a dense layer with 64 units, a dropout layer with a rate of 0.66 and a dense layer with 1 unit, respectively.

Output dimension: Model_(1,1) and model_(3Res,1) have an output dimension of 76x1, while the model_(5Res,1) has an output dimension of 38x1. This means that each respective model produces a prediction vector of either 76 or 38 in length.

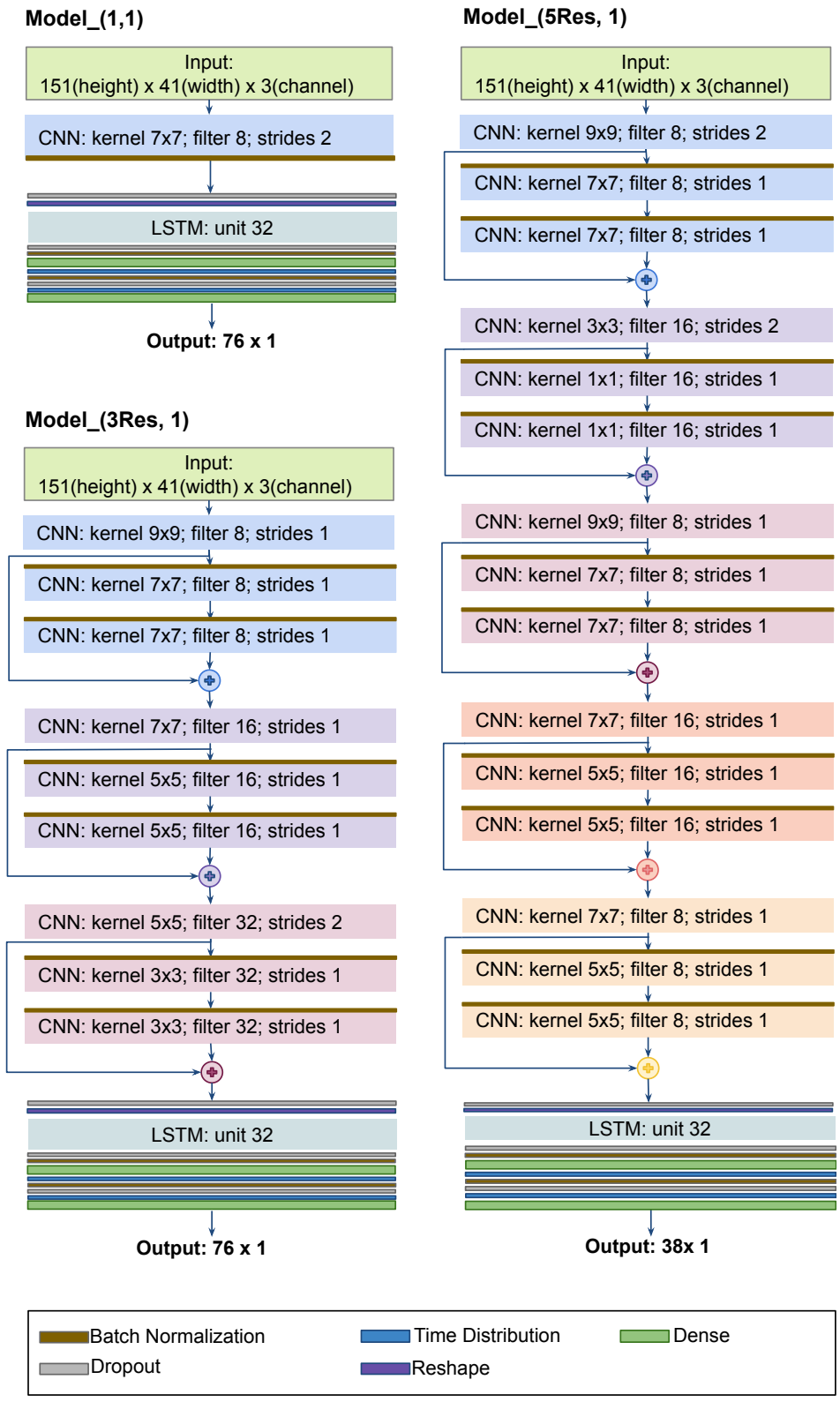

Figure 4: Selected Models: Based on the testing accuracy and the parameter count, we select three models. All three have a prediction vector accuracy of 99.5%. Model_(1,1) has a detection accuracy of 98.5% and 29k parameters; model_(5Res,1) has a detection accuracy of 96% and 73k parameters; and model_(3Res,1) has a detection accuracy of 99.5% and 151k parameters.

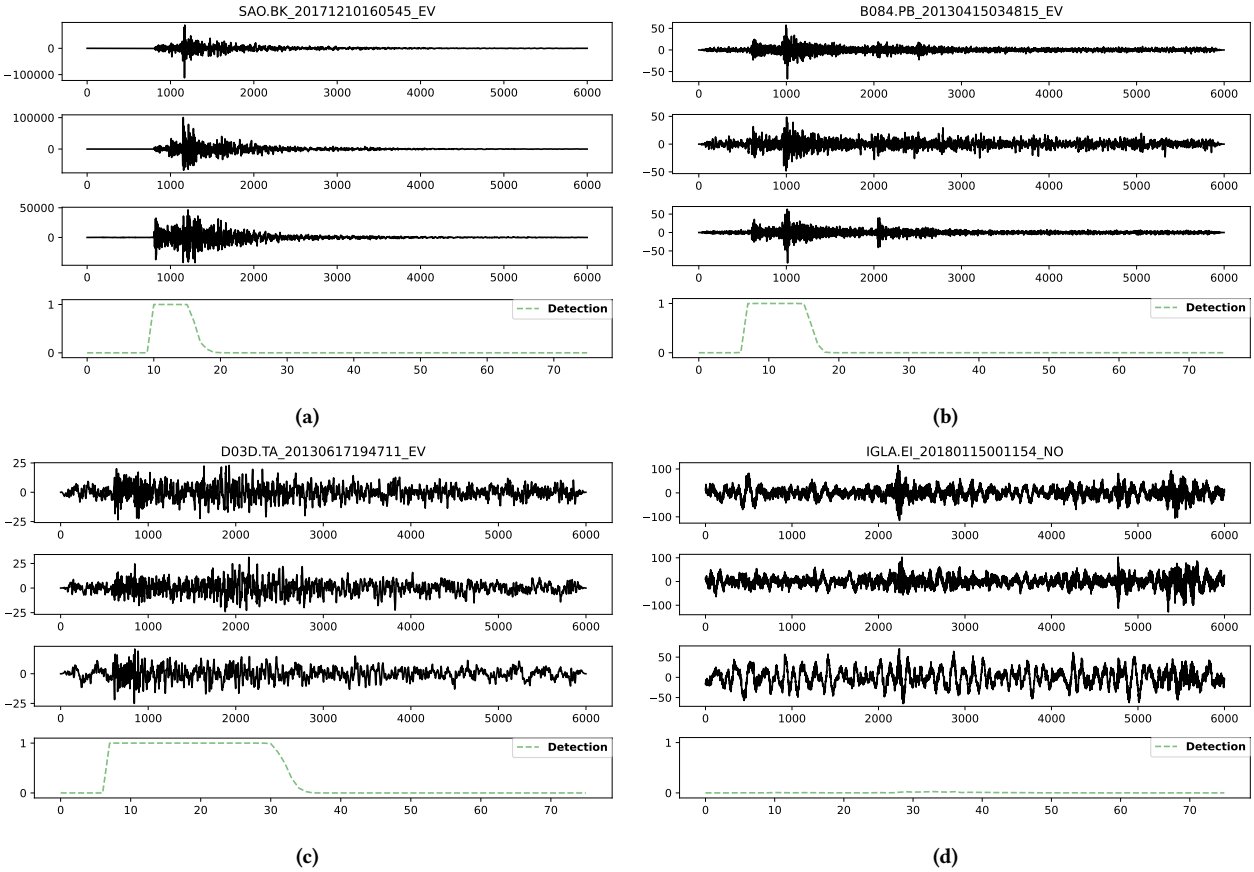

Figure 5: Visualization of selected results for LightEQ's model_(1,1)on the test set: (a) a seismogram with an earthquake with a magnitude of 2.43, a seismic source distance of 25.5 km, and an SNR of 50; (b) a seismogram with an earthquake with a magnitude of 0.5, a seismic source distance of 28 km, 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. [26].

We refer to the accuracy of the prediction vector as point-to-point accuracy, and we use this prediction vector to compute the integrated binary classification accuracy, which we refer to as detection accuracy. The three models have a similar point-to-point accuracy but differ in detection accuracy. This is due to the different output dimensions, and we discuss their impact in Section 5.7.

Epochs and learning rate: We train our models for 20, 30, and 30 epochs with batch sizes of 100, 100, and 80 for model_(1,1), model_(5Res,1), and model_(3Res,1), respectively. We set the learning rate to 0.001, as suggested by our hyperparameter search, and integrate an early stopping strategy to prevent overfitting. Additionally, we add a learning rate scheduler that starts with the recommended learning rate and reduces it by a factor of 10 when the learning rate metric stops improving for 5 epochs. This strategy ensures that the model can reach the minimum loss.

4.7 Quantization

To further reduce the model size and make it compatible with Cortex M4 microcontrollers, we perform post-training quantization and convert our 32-bit floating-point models to the nearest 8-bit fixed-point model with float fallback. Quantization generally reduces the model size at the cost of reducing model accuracy. This effect is evaluated in Section 5.4.

4.8 Implementation

We implement LightEQ for the Zephyr RTOS and use Tensorflow for training LightEQ-NN. To ensure that the model's weights do not monopolize all memory, we employ post-training quantization that is provided by TensorFlow, fixing weights to integer representations to reduce the model size. Our neural networks require between 150 kB to 400 kB of RAM memory and take 836 ms to 380082 ms for inference, depending on the chosen MCU, weights representation, and DSP configuration. We rely on Tensorflow Lite for Microcontrollers for on-device inference and utilize the DSPs of modern Cortex M4 for accelerated integer inference via CMSIS-NN.

While LightEQ is platform-independent and not bound to a specific microcontroller, we use the SoCs nRF52840 (nRF52) and nRF5340 (nRF53) in our evaluation. The nRF52 features a 64 MHz

Cortex-M4 CPU with FPU, 1 MB Flash, 256 kB of RAM and DSP instruction capabilities. The nRF53 has a dual Arm Cortex-M33 with FPU that can be clocked at either 128 or 64 MHz, 1 MB Flash, 512 kB RAM and DSP instruction capabilities.

5 EVALUATION

In this section, we experimentally evaluate the effectiveness of LightEQ and its three models. We compare precision, recall, and F1 score of both, the bit-8 integer and 32-bit float versions of our models, with state-of-the-art models. We analyze the impact of the selected neural network models in terms of number of parameters, memory usage, and model size. Finally, we evaluate the overall energy requirements of Model_(1,1) on nrf52 MCU with and without the DSP integer accelerator.

5.1 Data

We use the STanford EArthquake Dataset (STEAD) [25] to train our models. STEAD is a high-quality, large-scale dataset consisting of earthquake (with an epicenter distance of less than 350 km from the recording station) and non-earthquake (background noise) signals recorded by seismic instruments. Together, these data comprise 1,265,657 one-minute-long 3-component seismograms. Each seismogram has a recording length of 60 sec and is sampled with 100 Hz resulting in 6000 samples for each component per minute. In the dataset, approximately 1 million seismograms are labeled as earthquakes and 300k samples are labeled as non-earthquake.

To add more diversity to the training, we apply data aggregation to the training data [34]. Half of the data in the dataset is the augmented versions of the STEAD dataset. This increases the size of our dataset by a factor of 2. Data augmentation includes randomly adding Gaussian noise to the earthquake signal with a probability of 0.4 and randomly shifting the event within the trace with a 0.9 probability. For our extensive hyperparameter search, we use the whole training dataset with augmentation to see the robustness of our model search. This further reduces the likelihood of over-fitting.

5.2 Training and Testing

We randomly split our dataset into a training, validation, and test set of 85%, 5%, and 10%, respectively. We train the models on 4 NVIDIA TITAN X Pascal GPUs, each with 12 GB of memory. We use binary cross-entropy as our loss function and applied an Adam optimization algorithm.

To evaluate the detection performance of the models, we use 253,131 test samples. Output of models is a 76x1 or 38x1 dimensional probability vector. These probabilities are compared to upper and lower bound threshold values (tr) to integrate them into a binary classification. Finally, we use the confusion matrix to compute the performance of our models.

Accuracy is the standard performance measure which is a fraction of correct classifications – correctly classified earthquakes as true positive (TP) and noise as true negative (TN) – over the total number of classifications including also the falsely classified events – noise classified as earthquakes (FP) and earthquakes classified as noise (FN). Accuracy is computed as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision, on the other hand, refers to the correctness of the model when it detects an earthquake:

$$precision = \frac{TP}{TP + FP} \tag{2}$$

Recall is the fraction of true events correctly detected and is computed as:

$$recall = \frac{TP}{TP + FN} \tag{3}$$

The F1 score is a harmonic mean of precision and recall. In comparison accuracy, F1 gives a better measure of the incorrectly classified cases than accuracy, which measures correctly identified cases.

$$Fscore = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

5.3 Results

Before we compare LightEQ-NN to the state of the art, we conduct a series of benchmarks to evaluate the different models of LightEQ. Figure 5 shows four seismograms along with the event detection results of NN, using Model_(1,1) with 76 output data points for three different earthquakes (a) to (c) and one recording of background noise (d). Each of the seismic signals was recorded with three components (Z, N, and E), as seen in section 2. The earthquake magnitudes, source-receiver distances, 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. LightEQ-NN classifies all four examples correctly as earthquakes (regardless of their magnitude) or noise, respectively. We perform the classification of the 253,131 events of STEAD and achieve high precision, recall, and f1 scores for all three evaluated LightEQ-NN models, including the quantized versions, as seen in Figure 6.

5.4 Comparison with other Models

Figure 6 compares the performance of the earthquake signal detection using the LightEQ models (both float and int quantized) with the performance of a simple STA/LTA [17] as well as the state-of-the-art models CRED [26], EQTransformer [22], and LCANet [37] using the STEAD dataset. STA/LTA shows the lowest precision and F1 scores, while EQTransformer and LCANet show the best results, closely followed by CRED. The LightEQ Model_(3Res,1)$_{float}$ shows an on-par performance compared to EQTransformer and LCANet. Quantizing this model reduces its performance only slightly, but the quantized version still shows on-par results compared to CRED.

Model_(5Res,1)$_{float}$ has an on-par precision, recall and F1 score as CRED, while Model_(1,1)$_{float}$ as well as Model_(1,1)$_{int}$, perform on-par or even slightly better than CRED in terms of recall and F1 score. These results are significant, considering that CRED requires four times the number of parameters and has a 3.7 times model size compared to Model_(5Res,1)$_{float}$. When comparing CRED

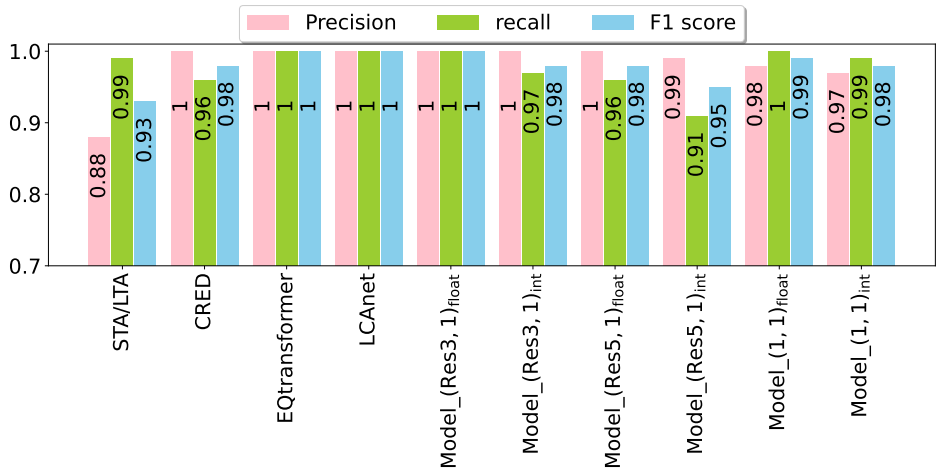

Figure 6: LightEQ-NN compared to state-of-the-art models: STA/LTA refers to a non-deep learning model, while CRED, EQTransformer, and LCANet are NN models. Model_(3Res,1), Model_(5Res,1) and Model_(1,1) are float and integer quantized versions of LightEQ-NN models. Overall, LightEQ-NN performs on par with CRED while slightly less good than EQTransformer and LCANet while being significantly more resource efficient.

Table 2: Float and integer quantized versions of LightEQ-NN and state-of-the-art models. Params denotes the number of model parameters. Note that EQTransformer and LCANet support phase picking next to earthquake detection, which is outside the scope of this paper. The table underlines the resource-efficiency of LightEQ when compared to the state of the art.

Model			Input size	Output size	Params	Model size	Detection	Phase picking
STA/LTA			(6000x3)	(1x1)			x	
CRED			(6000x3)	(38x1)	293k	1.1 MB	x	
EQTransformer			(6000x3)	(6000x1)	378k	1.4 MB	x	x
LCANet			(6000x3)	(6000x1)	210k	818 kB	x	x
LightEQ	Model_(1,1)	int	(6000x3)	(76x1)	29k	39 kB	x	
		float	(6000x3)	(76x1)	29k	126 kB	x	
	Model_(5Res,1)	int	(6000x3)	(38x1)	73k	94 kB	x	
		float	(6000x3)	(38x1)	73k	300 kB	x	
	Model_(3Res,1)	int	(6000x3)	(76x1)	151K	166 kB	x	
		float	(6000x3)	(76x1)	151K	601 kB	x	

with Model_(1,1)$_{int}$, CRED require 10 times more parameters and twenty-eight times the model size.

Table 2 shows the model characteristics and their resource demand. With only 29k, Model_(1,1) has the least number of parameters with an output dimension of 76x1, followed by Model_(5Res,1), which has 73k parameters with an output dimension of 36x1. Model_(3Res,1) is the largest evaluated LightEQ-NN model, but still, it has fewer parameters and a smaller model size compared to CRED [26], EQTransformer [22], and LCANet [37]. Quantizing the LightEQ-NN models to 8-bit integer fixed point with float fallback reduces the model size by a factor of three to four. Our integer quantized models have no floating-point operations and use the DSP of modern Cortex M4 microcontroller for optimization.

5.5 Energy Trace and Inference Time

An important aim of LightEQ is reduction of energy consumption. Figure 7 shows a representative comparison of the energy drawn for model_(1,1) with and without the use of the DSP on a nrf52840 MCU, along with the energy consumption for the STFT operation on a 1-minute time series. Our measurements show that the nrf52840 consumes 30.56 mJ during an inference of 5181 ms and 20 ms of OS operations with DSP disabled, whereas with DSP enabled, we measure an energy consumption of 5.86 mJ for 24 ms OS background work and 932 ms for the actual inference. For the other models and platforms, the results are comparable, with inference time being the dominating factor. We report inference time in Tables 3 and 4 for the respective platforms. These results highlight the positive impact of DSPs on the performance of inference of integer quantized models.

Table 3: Resource requirements and inference time for 8-bit integer and 32-bit float model for all three LightEQ models on nrf52840 MCU with CMSIS NN enabled and disabled.

nrf52840					
		without CMSIS NN		with CMSIS NN	
		int8	float32	int8	float32
model_(1,1)	exe time (ms)	5181	-	932	-
	RAM (bytes)	195688	-	198136	-
	Flash (bytes)	595064 = 56.75%	-	608152 = 58.00%	-
model_(5Res,1)	exe time (ms)	60588	-	3739	-
	RAM (bytes)	150184	-	151464	-
	Flash (bytes)	446536 = 42.58%	-	459624 = 43.83%	-
model_(3Res,1)	exe time (ms)	-	-	-	-
	RAM (bytes)	-	-	-	-
	Flash (bytes)	-	-	-	-

Table 4: Resource requirements and inference time for 8-bit integer and 32-bit float model for all three LightEQ models on nrf5340 MCU with CMSIS NN enabled and disabled

nrf5340					
		without CMSIS NN		with CMSIS NN	
		int8	float32	int8	float32
model_(1,1)	exe time (ms)	4531	3683	836	3629
	RAM (bytes)	195696	227984	198144	230368
	Flash (bytes)	597260 = 56.96%	488956 = 46.63%	610400 = 58.21%	502096 B = 47.88%
model_(5Res,1)	exe time (ms)	51379	48519	3469	47620
	RAM (bytes)	150176	210320	151456	210208
	Flash (bytes)	448732 = 42.79%	593340 = 56.59%	461872 = 44.05%	606480 = 57.84%
model_(3Res,1)	exe time (ms)	380082	-	20285	-
	RAM (bytes)	403472	-	407552	-
	Flash (bytes)	729820 = 69.60%	-	742960 = 70.85%	-

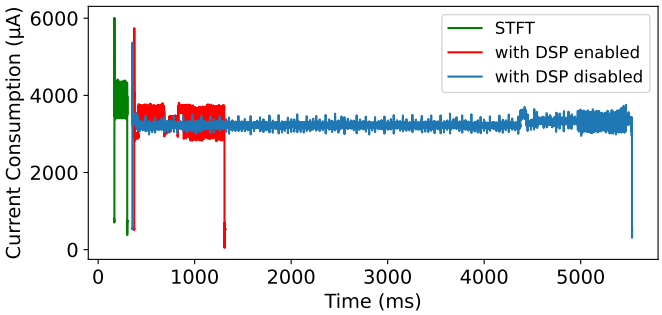

Figure 7: Energy measurements for model_(1,1) on nrf52840. STFT takes 138 ms and consumes 0.87 mJ of energy. It takes 932 ms and 5.86 mJ to perform inference with DSP enabled and 5,181 ms and 30.56 mJ with DSP disabled. These results underline the benefits of devising integer quantized models for modern DSP-enabled microcontrollers.

5.6 Resource Requirements

Tables 3 and 4 summarize the RAM and Flash consumption for all three LightEQ-NN models on both microcontrollers. Only the integer quantized model_(1,1) and model_(5Res,1) are deployable on the nrf52840 (see Table 3). Both models consume about 50% of the flash. Model_(1,1) and Model_(5Res,1), both with CMSIS enabled, require 76% and 58% of the RAM, respectively. Inference time for Model_(1,1) is reduced from 5 second to 1 second with CMSIS enabled, whereas inference time for Model_(5Res,1) is reduced from 60 to 4 seconds. The nrf5340 allows deploying all quantized models and even the 32-bit float versions of model_(1,1) and model_(5Res,1), while none of the state-of-the-art models are small enough to deployed on either of the microcontrollers, as shown in Table 2.

5.7 Discussion and Limitations

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

Output dimensions Adding CNN layers with a stride greater than 1 reduces the dimension of the prediction vector. This also reduces the number of parameters of the model, but may result in not detecting short earthquakes.

This is illustrated in Figure 8, which compares the accuracy of models with output dimensions of (151x1), (76x1), (38x1), and (19x1). Figure 8 (a) shows a seismogram of an earthquake from the STEAD dataset, the corresponding time-frequency map, and the prediction vectors of four models with output dimensions (151x1), (76x1), (38x1), and (19x1), respectively. This earthquake has a source-receiver distance of 34.81 km and a duration of about 20 seconds.

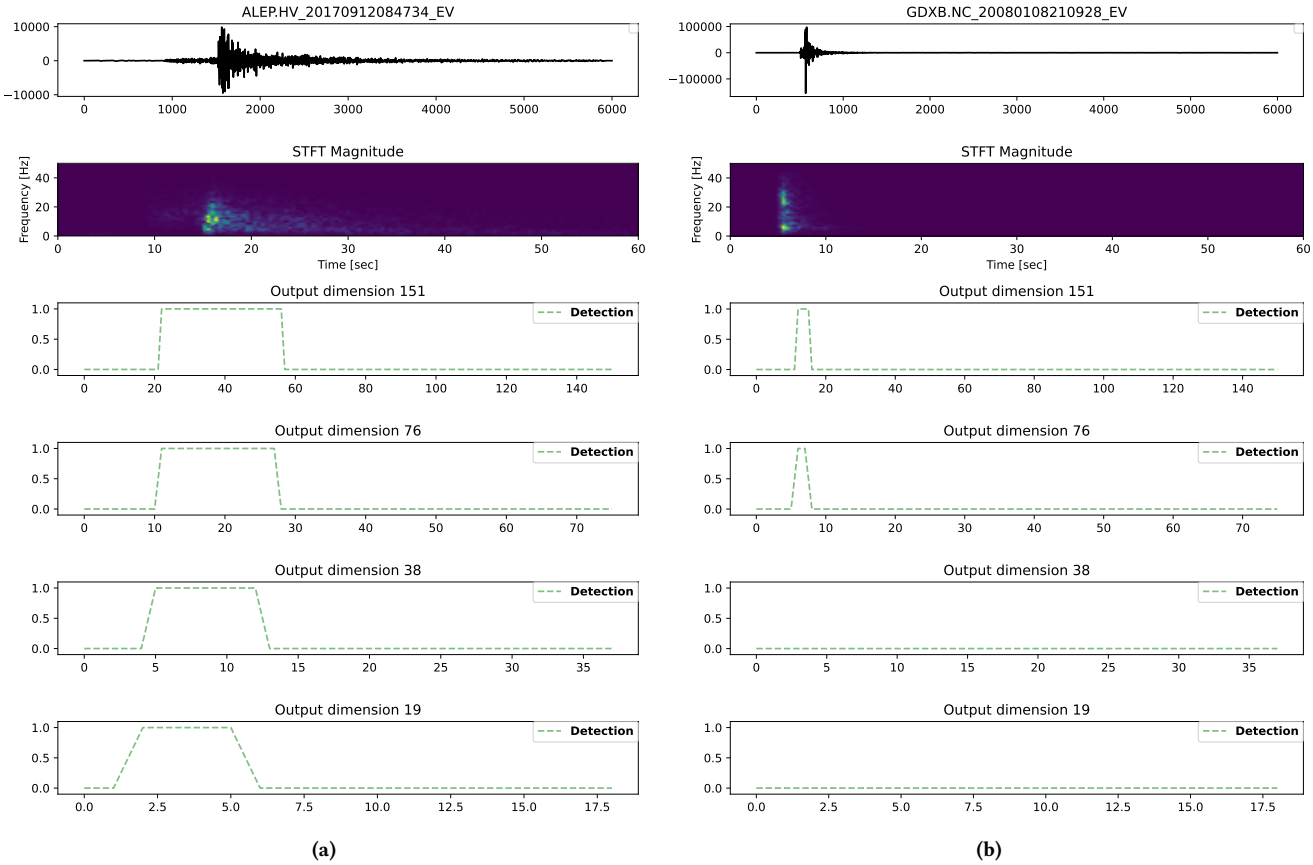

Figure 8: Classifications for output dimensions of (151x1), (76x1), (38x1), and (19x1): (a) A seismogram for a comparably long earthquake with a source-receiver distance of 34.81 km and event duration of about 20 seconds. Independent of the output dimensions, the earthquake is detected in all configurations. (b) A seismogram for a comparably short earthquake with a source-receiver distance of 2.05 km and event duration of only 4 seconds. For the shorter event the earthquake signal is not detected for the configurations of dimension 38 and 19.

Regardless of the output dimensions, the earthquake is detected in all configurations.

Figure 8 (b) shows a seismogram of a second, shorter earthquake, the corresponding time-frequency map and prediction vector for all four output dimensions. This earthquake has a source-receiver distance of 2.05 km and a duration of only about 4 seconds. As the length of the event in 8 (a) is comparably large, the earthquake signal is detected in all the outputs, while for the shorter event in (b), the earthquake signal is not detected for the configurations of dimension 38 and 19.

Overall, this illustrates the trade-off between the number of parameters and the model's accuracy. Reducing the output dimension of the models results in fewer parameters but may wrongly label small signals, hence learning on mislabeled data. If an application requires a setup that is sensitive to small earthquakes, one should consider using LightEQ with greater dimensions such as (151x1) or (76x1) as compared to (38x1) and (19x1). This can be viewed in Figure 9. For this experiment, we select four models with output dimensions of 151x1, 76x1, 38x1, 19x1. All these four models have a point-to-point accuracy of 99.5%. After classifying the output of the model into 0 or 1, the detection accuracy drops for models with output dimension 38x1 and 19x1.

Neural Network Architecture Search: The memory requirements of the model do not only depend on the number of parameters but on further aspects. For example, in Tensorflow Lite for MCUs, we have to unroll our RNNs to strictly structure them into feed-forward neural networks for compatibility reasons, which adds extra memory requirements. Moreover, the size of internal buffers depends on the size of the widest layers. Similarly, the exact impact of quantitation on model accuracy is unknown until actual quantization has been conducted. Such details need to be determined by experimentation and cannot be directly evaluated as part of the neural network architecture search. Thus, selecting models from the search requires a trail and error approach to determine which models have a small memory footprint.

6 CONCLUSION

In this paper, we introduce an efficient data processing pipeline and a set of lightweight deep-learning models for seismological data. The models are small enough to be deployed on microcontrollers, such Cortex M4. We optimize our model architectures consisting

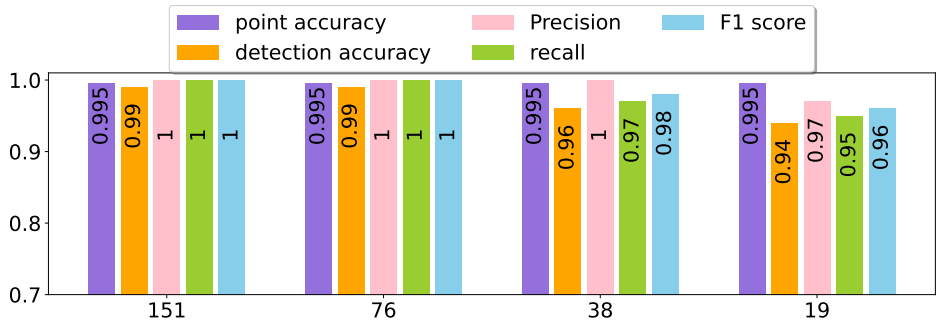

Figure 9: Point-to-point vs detection accuracy: All four models have a 99.5% point-to-point accuracy, but detection accuracy drops with the decrease in the output data points of the model.

of CNN and LSTM layers with an extensive hyperparameter search that varies kernel, filter, and stride sizes. The smallest model consists of 29k parameters, which is only a tenth compared to the 293k of the original CRED model. The quantized version has a model size of only 39 kB compared to 1.1 MB. This allows us to deploy the model on Cortex M4 microcontrollers, on which it has an inference time of 932 ms for one minute of raw data, an energy consumption of 5.86 mJ , and a flash requirement of 593 kB.

Our evaluation shows that it is possible to deploy our lightweight deep-learning models on constrained, embedded devices. This allows for a wide range of novel measurement and monitoring approaches in harsh environments. Potential applications include seafloor monitoring networks for marine subsurface operations as part of the transition of the energy system from fossil fuels to renewable energy sources, which includes the subsurface storage of CO_2 and the scale-up of offshore wind energy production. These operations require dense real-time monitoring, which may benefit from on-device event categorization and analysis. Further application might include monitoring submarine geo-hazards like landslides or volcanoes, for which seismological sensors play a crucial role. Especially, early warning systems would highly benefit from on-device analysis capabilities. Lightweight deep-learning models may also be implemented in seismological citizen science initiatives like MyShake [16], which currently applies resource and energy-demanding models. For example, lightweight and energy-efficient models may increase acceptance and the number of participants. While LightEQ is designed for seismological data, similar deep-learning models may also be applied to the on-device analysis of comparable continuous time series, such as acoustic recordings or pressure measurements.

ACKNOWLEDGMENTS

We wish to extend our heartfelt appreciation to the Helmholtz School for Marine Data Science (MarDATA) for their significant financial support [Grant No. HIDSS-0005] in facilitating our research efforts. Lastly, we express our gratitude to the anonymous reviewers whose valuable remarks and helpful suggestions greatly enhanced the caliber of this study.

REFERENCES

[1] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24 (2011).
[2] Jörg Bialas and Ernst R Flueh. 1999. Ocean bottom seismometers. *Sea Technol* 40, 4 (1999), 41–46.
[3] Ronald Newbold Bracewell and Ronald N Bracewell. 1986. *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York.
[4] 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.
[5] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. 2019. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 3009–3018.
[6] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al. 2021. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems* 3 (2021), 800–811.
[7] Gregory Giuliani, Hy Dao, Andrea De Bono, Bruno Chatenoux, Karin Allenbach, Pierric De 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 (2017), 222–233.
[8] Daniel 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.
[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.
[10] 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< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).
[11] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713.
[12] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713.
[13] Shengming Jiang. 2019. Marine internet for internetworking in oceans: A tutorial. *Future Internet* 11, 7 (2019), 146.
[14] Supriya Kapur, Asit Mishra, and Debbie Marr. 2017. Low precision rnns: Quantizing rnns without losing accuracy. *arXiv preprint arXiv:1710.07706* (2017).
[15] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. 2017. LSTM fully convolutional networks for time series classification. *IEEE access* 6 (2017), 1662–1669.
[16] Qingkai Kong, Young-Woo Kwon, Louis Schreierz, Steven Allen, Richard Allen, and Jennifer Strauss. 2015. Smartphone-based networks for earthquake detection. In *2015 15th international conference on innovations for community services (i4cs)*. IEEE, 1–8.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. 2012 AlexNet. *Adv. Neural Inf. Process. Syst* (2012), 1–9.
[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.
[19] 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 (2022), 848237.
[20] 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.
[21] JH Luetgert, DH Oppenheimer, and J Hamilton. 2010. The netquakes project-research-quality seismic data transmitted via the internet from citizen-hosted instruments. In *AGU Fall Meeting Abstracts*, Vol. 2010. S51E–03.
[22] 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), 1–12.
[23] S Mostafa Mousavi, Stephen P Horton, Charles A Langston, and Borhan Samei. 2016. Seismic features and automatic discrimination of deep and shallow induced-microearthquakes using neural network and logistic regression. *Geophysical Journal International* 207, 1 (2016), 29–46.
[24] Seyed Mostafa Mousavi and Charles Langston. 2016. Fast and novel microseismic detection using time-frequency analysis. In *SEG Technical Program Expanded Abstracts 2016*. Society of Exploration Geophysicists, 2632–2636.
[25] 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.
[26] 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), 1–14.
[27] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).
[28] Thibaut Perol, Michaël Gharbi, and Marine Denolle. 2018. Convolutional neural network for earthquake detection and location. *Science Advances* 4, 2 (2018), e1700578.
[29] 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.
[30] Sandra Sendra, Jaime Lloret, Jose Miguel Jimenez, and Lorena Parra. 2015. Underwater acoustic modems. *IEEE Sensors Journal* 16, 11 (2015), 4063–4071.
[31] BK Sharma, Amod Kumar, and VM Murthy. 2010. Evaluation of seismic events detection algorithms. *Journal of the Geological Society of India* 75, 3 (2010), 533–538.
[32] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306.
[33] Amadej Trnkoczy. 2009. Understanding and parameter setting of STA/LTA trigger algorithm. In *New Manual of Seismological Observatory Practice (NMSOP)*. Deutsches GeoForschungsZentrum GFZ, 1–20.
[34] David A Van Dyk and Xiao-Li Meng. 2001. The art of data augmentation. *Journal of Computational and Graphical Statistics* 10, 1 (2001), 1–50.
[35] Yue Wu, Youzuo Lin, Zheng Zhou, David Chas Bolton, Ji Liu, and Paul Johnson. 2018. DeepDetect: A cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing* 57, 1 (2018), 62–75.
[36] 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.
[37] 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), 4024–4035. https://doi.org/10.1109/JSYST.2021.3114689