engineering proceedings

MDPI

Proceeding Paper

Embedded Acoustic Sensing for Infant Cry Classification and Intelligent Comfort Response †

Jyoti Mishra and Timothy Malche *

Department of Computer Applications, Manipal University Jaipur, Jaipur 303007, Rajasthan, India; jyoti.191018608@muj.manipal.edu

* Correspondence: timothy.malche@jaipur.manipal.edu

† Presented at the 13th International Electronic Conference on Sensors and Applications, 18–20 November 2026; Available online: https://sciforum.net/event/ECSA-13.

Abstract

Most baby monitors transmit audio or video but cannot understand or respond to a baby's cry. We propose a low-power sensor node that uses TinyML to classify infant cries, play a soothing lullaby, and send alerts to caregivers without cloud connectivity. A micro-electro-mechanical system (MEMS) microphone feeds 1 s windows from which Mel-frequency cepstral coefficient (MFCC) features are extracted. A quantized INT8 neural network classifies cry, background noise and speech. It achieved 86.5% accuracy, 0.883 macro F1 and 0.96 area under the receiver operating characteristic (ROC) curve, with 69 ms latency, 8.8 kB RAM and 60.4 kB flash.

Keywords: TinyML; MEMS sensors; Good Health and Well-Being (UN SDG 3); environment monitoring

1. Introduction

Watching a sleeping baby all the time is not possible, so parents use baby monitors. Almost every monitor on the market works in the same way. A microphone picks up sound in the room, and a radio sends that sound to a phone or a base unit. The device itself does not understand anything. It cannot tell a cry from a passing truck, and it cannot do anything about what it hears.

This causes three problems. First, the parent is disturbed by every loud sound, not only by the ones that matter. Second, streaming audio all the time uses power and network bandwidth, so the battery does not last. Third, sending live audio from a bedroom to a phone or a cloud server is a privacy risk that many families do not want.

A sensor node that makes decisions by itself solves all three. Only a short message leaves the device, the radio stays off almost all the time, and no recording of the room is ever sent. To do this, we need a model small enough to run on a microcontroller. This is the field of TinyML [1]. The work also supports United Nations Sustainable Development Goal 3, Good Health and Well-Being, by making reliable infant monitoring cheap enough and private enough to use in homes that would not accept a cloud-connected camera.

There is already a lot of work on infant cry analysis. Review papers cover the features, models, and datasets in detail [2,3]. Recent studies use deep models such as ResNet and EfficientNet for cry detection [4], and some build small models for low-power hardware [5,6]. Almost all of this work trains and tests on one dataset. We found that this hides a serious problem, and measuring it is the main methodological idea of this paper.

Check for updates

Academic Editor: Jean-Marc Laheurte

Published: 3 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

The problem is the recording style. A dataset is not only a set of sounds. It is also a set of microphones, sample rates, file formats, and rooms. Suppose every cry in the training data was recorded on a phone, and every non-cry sound came from a sound effects library. A model can then get a high score by learning which dataset a clip came from. It never has to learn what a baby sounds like. The author of the ESC-50 dataset gives a warning about exactly this kind of hidden class information [7], and cross-dataset shift is now an active concern in embedded audio work [8,9].

2. Sensor Node

2.1. Choice of Board

We built the node on a Seeed Studio XIAO ESP32S3 Sense (Seeed Technology Co., Ltd., Shenzhen, China). The board was chosen because it already contains almost everything this application needs, so no custom analogue circuit had to be designed. ESP32-class boards (Seeed Technology Co., Ltd., Shenzhen, and China) are now a common target for TinyML audio work [10].

Most importantly for us, the Sense version has a digital micro-electro-mechanical system (MEMS) microphone (Seeed Technology Co., Ltd., Shenzhen, and China) built onto the board. It sends pulse density modulation (PDM) data straight into the inter-IC sound (I2S) peripheral of the ESP32-S3 on two pins. No preamplifier, no analogue filter, and no external ADC are needed. A MEMS part is also the right choice for a device that will sit in a nursery for months because its sensitivity does not drift with age or temperature the way an electret capsule can.

Figure 1 shows the board used in this work.

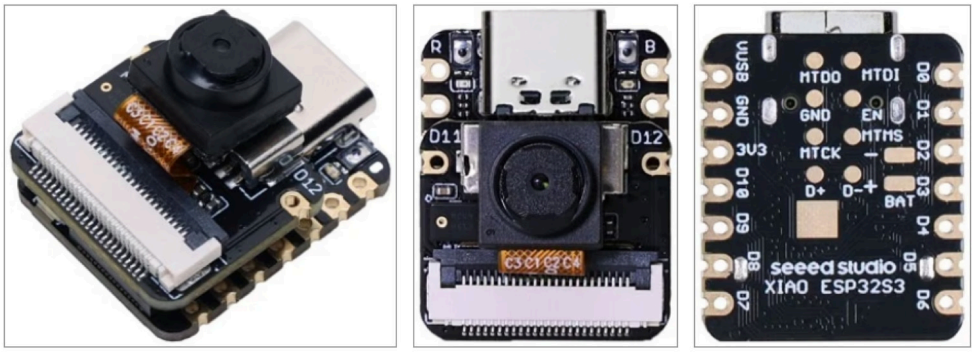

Figure 1. The Seeed Studio XIAO ESP32S3 Sense. (**a**) The board with the Sense expansion attached. (**b**) Top view. (**c**) Underside, showing the pin layout. The camera module visible in (**a**) and (**b**) is detachable and is not used in this work.

2.2. Node Design and Response Path

The microphone and a temperature and humidity sensor feed the board. The board drives a small speaker, sends Bluetooth Low Energy (BLE) alerts, and writes to the microSD card.

The temperature and humidity sensor is not part of the classifier. It only adds context, so that a saved event says what the room was like when the baby cried. Each event takes about 64 bytes, so months of events fit easily on the card.

The response is local first. The lullaby starts on the device before any message is sent, so the baby is soothed even if the parent's phone is off or out of range. The BLE message is 20 bytes long and holds the predicted class, the confidence, the time, the temperature, and the humidity. It holds no audio.

https://doi.org/10.3390/engproc2026163072

Two design choices keep the average power low. The radio sleeps except during an alert, and the card is written only when an event happens. A cooldown timer of 30 s stops one long crying spell from sending a stream of messages.

3. Materials and Methods

3.1. Training Data

We built the training set from three public datasets that were recorded independently of each other. Donate-a-Cry is a collection of infant cries recorded on smartphones by volunteers [11]. ESC-50 holds 2000 environmental sound clips in 50 classes, taken from the Freesound library [7]. MS-SNSD provides household noise and clean speech under a permissive license [12]. Reviews of environmental sound classification give a wider view of these datasets and the models used with them [13,14]. Table 1 shows what each one gives us.

Table 1. The three datasets used for training and testing.

Dataset	License	Original Rate	Class	Clips	Groups	Minutes
Donate-a-Cry	ODbL 1.0	8 kHz	cry	904	218	75.3
ESC-50	CC BY-NC 3.0	44.1 kHz	cry	40	18	3.3
ESC-50	CC BY-NC 3.0	44.1 kHz	noise	540	489	45.0
MS-SNSD	MIT	16 kHz	noise	537	24	44.8
MS-SNSD	MIT	16 kHz	speech	420	53	35.0
Total	-	-	3 classes	2441	802	203.4

Apart from the two classes, cry and noise, a third class, speech, is also used. This class consists of adult speech from the MS-SNSD dataset. Adult speech is the most common sound in a nursery after a baby. Without a separate class for it, every human voice pushes the model towards cry, and the reported cry precision becomes meaningless in a room where people talk. Speech comes from only one dataset.

3.2. Audio Preprocessing

Mixing datasets only works if every file goes through the same steps. Our pipeline has six steps: read the file, mix down to mono, resample to 8 kHz, cut into 5.000 s clips, drop clips quieter than -54 dBFS, and scale the rest to -20 dBFS RMS with a limit on the peak.

3.3. Shortcut Learning and Dataset Bias

To measure bias, we trained a simple logistic regression on a 26-number summary of each clip, made from the mean and standard deviation of 13 MFCCs, using grouped cross-validation. Cross-dataset shift of this kind is a known problem for TinyML audio models [9], and source-isolated evaluation has recently been proposed as a remedy [8]. This gives two numbers.

The first is how well the model can tell which dataset a clip came from. This reached 70.1%, against 33.3% for random guessing. That is high, as expected, because the three datasets really do sound different. On its own, it is not a fault.

The second is the source-identity baseline: the accuracy of a model that sees only the dataset name and always predicts that dataset's most common label. Measured on the same windows and the same split used for every other result in this paper, it scores 83.3% accuracy, against 45.2% for always guessing the majority class. This is the number that matters because any model scoring at or below it has not been shown to use the audio at all.

3.4. Features and Train/Test Split

We use Mel-frequency cepstral coefficient (MFCC) features. Each 1000 ms window at 8 kHz gives 50 frames of 13 coefficients, so the model input is 650 values. Windows move forward 500 ms at a time, giving 9 windows per 5 s clip, and 21,969 windows in total.

The split is grouped, not random. Donate-a-Cry has up to 13 clips from the same phone, and ESC-50 has up to 7 clips cut from one original upload. A random split would put near-identical clips on both sides and give a falsely high score. We therefore split on the group key of Table 1, giving 1963 training clips (17,667 windows) and 478 held-out test clips (4302 windows), with no group on both sides. The same key is sent to the training platform as metadata, so its internal validation split follows the same rule. All results in Section 4 are on that validation split of 3533 windows.

3.5. Model and Training

The model is a one-dimensional convolutional neural network (CNN) with three convolution blocks of 16, 32, and 64 filters, kernel size 3, each followed by max pooling and 25% dropout, then a dense SoftMax layer for three classes. We trained it with Adam for 100 epochs, learning rate 0.001, batch size 32, on the Edge Impulse platform [15]. Table 2 lists the settings.

Table 2. Feature and model settings.

Stage	Configuration
Input	Window/step: 1000/500 ms; sampling: 8 kHz
MFCC	13 coefficients; frame/step: 20/20 ms; 32 Mel filters; 256-point FFT; range: 300–4000 Hz; 650 values/window
Model	Conv. filters: 16, 32, 64; kernel: 3; dropout: 0.25/block
Training	Adam; categorical cross-entropy; 100 epochs; batch: 32; 55,200 updates; noise and time/frequency masking
Deployment	8-bit per-channel quantization

3.6. Baseline Models

To check that the CNN is worth its complexity, we trained nine baseline models on the same audio, using 650-value MFCC windows and a grouped 80/20 split of the same 1963 training clips (13,509 training windows from 512 groups against 4158 validation windows from 129 groups, with no group shared). The baselines are a majority-class predictor, the source-identity baseline in Section 3.3, Gaussian naive Bayes, k-nearest neighbors, a decision tree, logistic regression, a linear support vector machine, a random forest of 200 trees, and a 128-64 multilayer perceptron. All inputs were standardized before fitting. Three differences should be noted: the baselines use an independent grouped 80/20 split while the CNN uses the training platform's own grouped validation split of the same clips, the baseline features come from a reference MFCC implementation without the platform's pre-emphasis and normalization window, and the baselines are trained in floating point while the reported CNN result is the quantized model.

4. Results

4.1. Training Graphs

Figure 2 shows accuracy and loss for each epoch. Both lines rise quickly for the first 20 epochs and then level off. After about epoch 30, a small gap opens between the two sets and slowly widens; training accuracy ends near 0.876 while validation ends near 0.851, and training loss ends near 0.303 against 0.354 for validation. This is mild overfitting. It does

not harm the result because validation accuracy never falls and validation loss is still slowly decreasing at epoch 100.

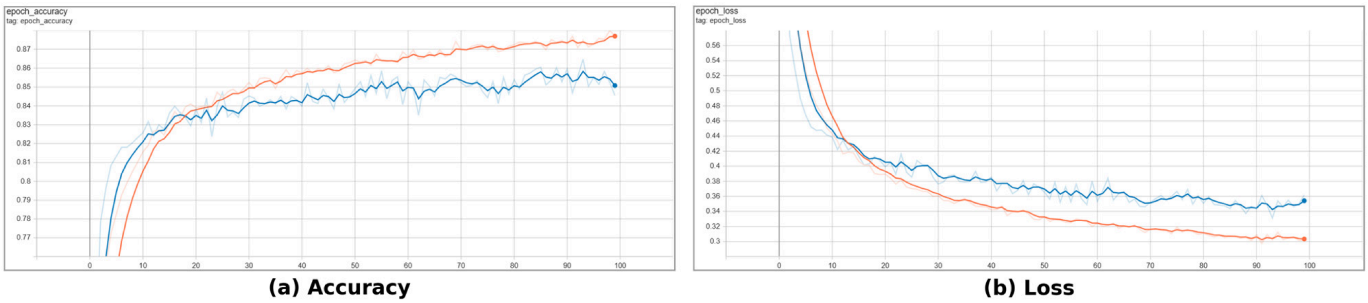

Figure 2. Training graphs over 100 epochs, shown in two columns. (**a**) Accuracy. (**b**) Loss. In both panels, the orange line is the training set and the blue line is the validation set.

4.2. Classification Results

The 8-bit model reaches 86.5% accuracy with a loss of 0.33 and a receiver operating characteristic area under the curve (ROC AUC) of 0.96. Weighted precision is 0.87, weighted recall is 0.87, and weighted F1 is 0.86. Figure 3 and Table 3 give the per-class detail.

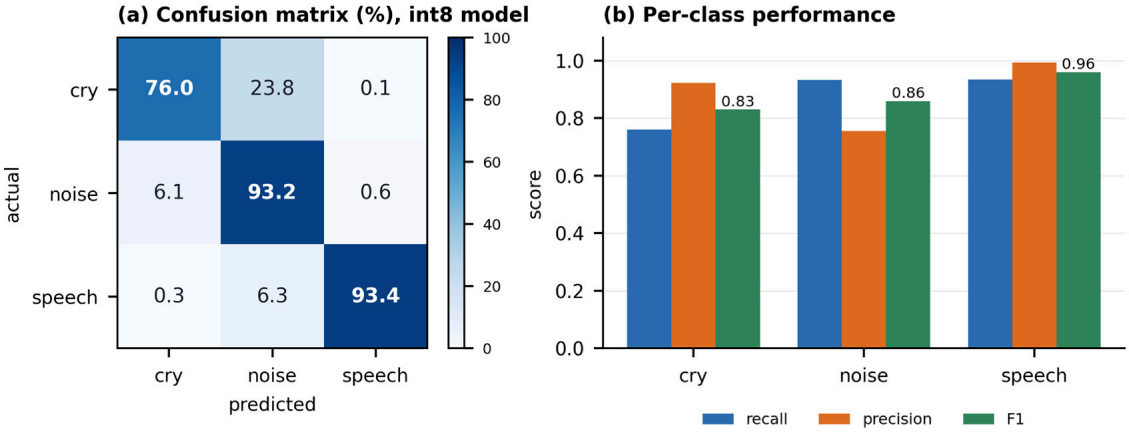

Figure 3. Results for the 8-bit model on the validation set. (**a**) Confusion matrix in percent. (**b**) Precision, recall, and F1 for each class.

Table 3. Results for each class. Precision is calculated from the confusion matrix in Figure 3a and the class counts of the validation split.

Class	Precision	Recall	F1	Main Error
cry	0.92	0.76	0.83	23.8% called noise
noise	0.80	0.93	0.86	absorbs errors from both classes
speech	0.98	0.93	0.96	6.3% called noise
macro average	0.898	0.876	0.883	-

4.3. Metrics Beyond Accuracy

Accuracy on its own is misleading here because the three classes are not balanced and because the source-identity baseline already reaches 83.3% accuracy without using the audio. Table 4 therefore reports five further measures. Balanced accuracy averages the per-class recalls, so a class with few samples counts as much as a large one. The Matthews correlation coefficient and Cohen's kappa both correct for agreement that would happen by chance, and both are near zero for a model that only guesses the majority class.

Table 4. Evaluation metrics for the 8-bit model on the validation set.

Metric	Value	What It Tells Us
Accuracy	86.5%	overall correct fraction
Balanced accuracy	87.6%	average of the three class recalls
Macro precision	0.898	all classes weighted equally
Macro recall	0.876	all classes weighted equally
Macro F1	0.883	the headline number for this task
Weighted F1	0.86	classes weighted by size
Matthews correlation coefficient (MCC)	0.789	chance-corrected agreement
Cohen's kappa	0.783	chance-corrected agreement
ROC AUC	0.96	ranking quality, threshold-free

Accuracy, weighted F1 and ROC AUC are as reported by the training platform. The remaining measures were computed from the confusion matrix in Figure 3a together with the class counts of the validation split; this computation reproduces the reported accuracy and all three per-class F1 values.

4.4. Comparison with Baseline Models

Table 5 compares the CNN against the nine baseline models in Section 3.6.

Table 5. The proposed model against nine baseline models, all evaluated under the same grouped protocol on the same audio.

Method	Accuracy (%)	Macro F1	MCC
Majority class	45.2	0.208	0.000
Source identity only	83.3	0.614	0.752
k-nearest neighbors (k = 5)	55.8	0.408	0.267
Linear support vector machine	64.6	0.616	0.423
Logistic regression	65.4	0.633	0.439
Decision tree	65.9	0.626	0.449
Gaussian naive Bayes	76.0	0.755	0.626
Multilayer perceptron (128-64)	81.4	0.816	0.702
Random forest (200 trees)	85.9	0.863	0.775
1D CNN, 8-bit (this work)	86.5	0.883	0.789

Three findings come out of this. First, the CNN gives the best macro F1 of all ten methods, but its accuracy is only 0.6 points above a random forest. Anyone reporting accuracy alone would conclude that the two are equal. The macro F1 gap of 0.020 and the MCC gap of 0.014 show that the CNN is better distributed across the three classes.

Second, the linear methods do badly. Logistic regression and a linear support vector machine both land in the mid-sixties, far below the source-identity baseline. The classes are not linearly separable in raw MFCC space, which is a useful negative result; the convolution structure is doing real work and is not merely a fashionable choice.

Third, and most important for deployment, the random forest that comes closest in accuracy uses 200 trees and cannot be placed on a microcontroller with 8.8 kB of working memory. The CNN wins on the metric that matters and is the only competitive method that actually fits on the device.

4.5. On-Device Performance

The quantized model was compiled for the target and profiled. Table 6 gives the measured figures.

Table 6. Measured on-device performance on the XIAO ESP32S3 Sense.

Measure	Value	Share of the Board
Inference time per window	69 ms	13.8% duty cycle at a 500 ms step
Peak RAM	8.8 kB	about 0.1% of the 8 MB PSRAM
Flash	60.4 kB	about 0.7% of the 8 MB flash
Model format	8-bit integer, per channel	-

https://doi.org/10.3390/engproc2026163072

4.6. From Window Predictions to Event Detection

A cry does not last one second. It lasts many seconds and covers many windows, so the prediction should be smoothed over a run of windows instead of taken from a single one. We require M out of N windows in a row to be classified as a cry, with $N = 9$, the number of windows in a 5 s block. Treating the windows as separate gives the graphs in Figure 4 and the values in Table 7.

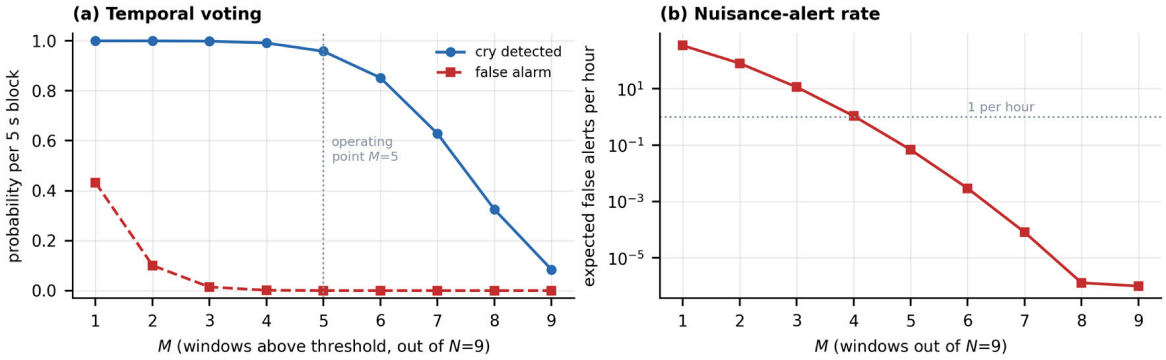

Figure 4. Temporal smoothing. (**a**) Event-level recall and false positive rate per 5 s block against M. (**b**) Expected false alerts per hour, log scale.

Table 7. Effect of temporal smoothing, using a per-window cry recall of 0.760 and a noise-to-cry error rate of 0.061.

M of 9	Event-Level Recall	False Positive Rate	False Alerts per Hour
1	1.000	4.32×10^{-1}	346
2	0.9999	1.01×10^{-1}	80.5
3	0.9990	1.44×10^{-2}	11.5
4	0.9919	1.36×10^{-3}	1.09
5	0.9584	8.64×10^{-5}	0.069

We recommend $M = 5$. It recovers 95.8% of cry blocks and produces 0.069 false alerts per hour, which is about one every fourteen hours, or roughly one false wake-up every other night. Setting $M = 4$ raises detection to 99.2% but allows 1.09 false alerts per hour, which over an eight-hour night means nearly nine false alarms. For a device meant to let a parent sleep, that trade is clearly wrong. A per-window cry recall of 76.0% becomes an event-level recall of 95.8% without changing the model at all, only the decision rule.

5. Discussion

5.1. Where the Errors Come from

The largest error is a cry being classified as noise, at 23.8%. Much of this is label noise rather than model failure. The Donate-a-Cry recordings were made on smartphones in real homes, so a cry is usually mixed with whatever else was happening in the room, for example, a television, other voices, traffic, or a washing machine. A baby also cries in bursts, with pauses for breath between them. The label belongs to the whole 5 s clip and is then inherited by all nine of its 1 s windows, so a window that lands in a pause, or in a stretch where the background is louder than the infant, is labeled a cry while containing little or no cry.

We measured how often this happens. Using the fraction of frames with a clear pitch as a simple measure of how cry-like a window is, cry windows have a median of 0.429 against 0.184 for noise windows, so the two classes are well separated on average. But 16.0% of cry windows fall below 0.15, and those have a median of 0.061, which is lower than the median noise window. About one cry window in six therefore carries the cry label while

sounding, by this measure, less voiced than ordinary background noise. That accounts for roughly two thirds of the 23.8% cry-to-noise error, and no classifier could separate those windows from noise using the audio alone.

This is also the strongest argument for the temporal smoothing stage of Section 4.6. The mislabeled windows are scattered through a cry episode rather than grouped together, so a decision taken over nine consecutive windows steps over them. This is why a per-window recall of 76.0% becomes an event-level recall of 95.8% without any change to the model.

5.2. Novelty

The individual parts of this system are established: MFCC features, a small convolutional network, 8-bit quantization, and a standard embedded deployment path. We do not claim any of them as new. What we do claim is that the way these parts are usually evaluated in the infant cry literature is not sound, and that our evaluation protocol is the contribution.

Three specific practices carry that claim: building the training set so that every class comes from at least two independently recorded datasets; splitting on recording groups rather than at random, which matters because a single phone contributes up to 13 clips; and reporting the source-identity baseline next to accuracy, so a reader can see how much of the score is explained by recording style alone. On our data, that baseline is 83.3% in accuracy but only 0.614 in macro F1, and the CNN reaches 86.5% and 0.883. A study using one dataset per class would have had a baseline of 100% and no way to tell.

5.3. Limitations

Four limitations should be stated clearly. First, the speech class comes from one dataset only, so the dataset and label are still mixed together for that class. Its F1 of 0.96 is therefore optimistic. Second, all results here are on the validation split. The held-out test set of 478 clips has not been scored yet, and that is the number a final claim should use.

Third, the alert delivery time and the battery life have not yet been measured. Inference time, RAM, and flash were measured and are given in Table 6, but the end-to-end time from cry to phone notification depends on the BLE connection interval and has not been timed under load. The smoothing figures also rest on an assumption that windows are separate, which only a real test can check. Fourth, none of the training audio was recorded in a real nursery. A small test-only set recorded on the XIAO microphone (Seeed Technology Co., Ltd., Shenzhen, China), in the room where the node will be used, would turn a laboratory result into a deployment result.

6. Conclusions

We have described a small sensor node that detects infant cries on the device, plays a lullaby, alerts a parent over Bluetooth Low Energy, and saves each event with room temperature and humidity, without sending any audio. It is built on a XIAO ESP32S3 Sense board, whose built-in digital microphone removes the need for any analogue front-end design. The 8-bit model reaches 86.5% accuracy, 0.883 macro F1, 0.789 Matthews correlation and 0.96 ROC AUC on a grouped split of 203.4 min of audio drawn from three public datasets and runs in 69 ms using 8.8 kB of RAM and 60.4 kB of flash.

Author Contributions: Conceptualization, T.M.; methodology, J.M., T.M.; software, J.M.; validation, J.M.; formal analysis, J.M.; investigation, J.M.; writing—original draft preparation, J.M.; writing—review and editing, T.M.; visualization, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All audio used in this study is publicly available. Donate-a-Cry is released under ODbL at https://github.com/gveres/donateacry-corpus (accessed on 17 March 2026). ESC-50 is distributed under CC BY-NC 3.0 at https://github.com/karolpiczak/ESC-50 (accessed on 17 March 2026). MS-SNSD is distributed under the MIT license at https://github.com/microsoft/MS-SNSD (accessed on 17 March 2026). The notebook that reproduces the dataset build, the bias check, and every figure is available from the corresponding author. Board documentation for the Seeed Studio XIAO ESP32S3 Sense is available at https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/ (accessed on 17 March 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Elhanashi, A.; Dini, P.; Saponara, S.; Zheng, Q. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. *Electronics* **2024**, *13*, 3562. [CrossRef]
2. Ji, C.; Mudiyanselage, T.B.; Gao, Y.; Pan, Y. A Review of Infant Cry Analysis and Classification. *EURASIP J. Audio Speech Music Process.* **2021**, *2021*, 8. [CrossRef]
3. Hashemi, S.M.H.; Kolivand, H.; Khan, W.; Saba, T. Infant Cry Analysis: A Survey of Datasets, Features, and Machine Learning Techniques. *IEEE Trans. Affect. Comput.* **2025**, *in press*. [CrossRef]
4. Herlea, D.M.; Iancu, B.; Ardelean, E.R. A Study of Deep Learning Models for Audio Classification of Infant Crying in a Baby Monitoring System. *Informatics* **2025**, *12*, 50. [CrossRef]
5. Eldine, O.B.; Selim, M.; El-Makky, N.; Ghanem, N.M. TinyML Prototype of Infant Cry Classification System. *IEEE Access* **2025**, *in press*. [CrossRef]
6. Yu, H.; Li, Y. Infant Cry Detection in Noisy Environment Using Blueprint Separable Convolutions and Time-Frequency Recurrent Neural Network. *arXiv* **2025**, arXiv:2508.19308. [CrossRef]
7. Piczak, K.J. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia, Brisbane, Australia, 26–30 October 2015*; ACM Press: New York, NY, USA, 2015; pp. 1015–1018. [CrossRef]
8. Wang, Z.; Yu, G. A Sensor-Based TinyML Acoustic Monitoring System for Edge-Side Animal Sound Recognition on Resource-Constrained Microcontrollers. *Sensors* **2026**, *26*, 3972. [CrossRef] [PubMed]
9. Trigkas, A.; Piromalis, D.; Papageorgas, P. A Federated Approach for Adaptive Urban Sound Classification on TinyML Edge Devices. *Sensors* **2026**, *26*, 2854. [CrossRef] [PubMed]
10. Gonzalez, M.; Gutierrez, S.; Espinosa, R.; Ponce, H. Deploying Real-Time Speech Recognition on ESP32 Using TinyML and Edge Impulse. In *Artificial Intelligence—COMIA 2025; Communications in Computer and Information Science*; Springer: Cham, Switzerland, 2025; Volume 2552. [CrossRef]
11. Veres, G. Donate-a-Cry Corpus. Available online: https://github.com/gveres/donateacry-corpus (accessed on 29 August 2026).
12. Reddy, C.K.A.; Beyrami, E.; Pool, J.; Cutler, R.; Srinivasan, S.; Gehrke, J. A Scalable Noisy Speech Dataset and Online Subjective Test Framework. In *Proceedings of Interspeech 2019, Graz, Austria, 15–19 September 2019*; ISCA: Baixas, France, 2019; pp. 1816–1820. [CrossRef]
13. Mohaimenuzzaman, M.; Bergmeir, C.; West, I.T.; Meyer, B. Environmental Sound Classification on the Edge: A Pipeline for Deep Acoustic Networks on Extremely Resource-Constrained Devices. *Pattern Recognit.* **2023**, *133*, 109025. [CrossRef]
14. Bansal, A.; Garg, N.K. Environmental Sound Classification: A Descriptive Review of the Literature. *Intell. Syst. Appl.* **2022**, *16*, 200115. [CrossRef]
15. Hymel, S.; Banbury, C.; Situnayake, D.; Elium, A.; Ward, C.; Kelcey, M.; Baaijens, M.; Majchrzycki, M.; Plunkett, J.; Tischler, D.; et al. Edge Impulse: An MLOps Platform for Tiny Machine Learning. In Proceedings of the 6th MLSys Conference, Miami Beach, FL, USA, 4–8 June 2023. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.