TinyML Challenge 2022: Smart Weather Station Aurora

Final report of team CSEM

Jonathan Reymond, Robin Berguerand, Ayman Manzoor, Jona Beysens
{jonathan.reymond,robin.berguerand,ayman.manzoor,jona.beysens}@csem.ch

ABSTRACT

We present Aurora, a low-power and robust device measuring the humidity, the temperature, the pressure as well as the wind and rain level using Tensorflow Lite Micro and the real-time operating system (RTOS) μ111. The device has no mechanical moving parts and has a total cost less than 100 euros. The enclosure is 3D printed and consists of 4 parts that are easy to mount. The machine learning (ML) model to predict the wind and rain level processes directly the raw audio signal by using convolutional layers, max pooling layers and dense layers. When quantized, it fits into the available 128 KB of SRAM and 512 KB of flash memory on the used embedded board MAX78000FTHR. This low-power board has a built-in MEMS microphone and is connected to the BME280 sensor which measures the humidity, temperature and pressure in the environment. Extensive experiments are performed to design the enclosure, acquire a real-world dataset, analyze the collected data, as well as to train and evaluate a tiny machine learning model.

Figure 1: The smart weather station Aurora.

1 INTRODUCTION

When studying the distribution of weather stations around the world, one can observe that it is highly condensed in developed countries, but mostly nonexistent in developing countries. Although weather stations could clearly help these societies, for instance for farming, most of them have three potential issues :

(1) Energy consumption
(2) Price
(3) Robustness

These reasons motivated the TinyML foundation to launch a challenge: construct a low-cost and low-power weather station that has no mechanical parts that could break and difficult to replace.

2 PROTOTYPE

The high level idea of the prototype consists of measuring in one hand the temperature, humidity and pressure with one dedicated sensor, and on the other hand to record the environmental sound to infer the rain level by using a TinyML model, all embedded in a low-power microcontroller powered by a lithium battery as shown in the diagram in Fig. 2.

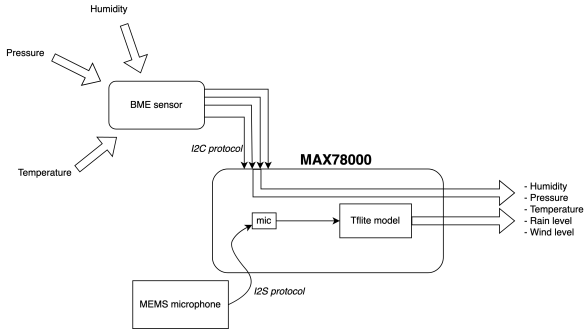

Figure 2: Aurora's architecture.

2.1 Electronics

2.1.1 Microphone. The MAX78000FTHR board has a MEMS microphone which directly integrated into the board. It has a large frequency range (from 50Hz up to 15KHz) and communicate via the I2S protocol. Since it produces directly a digital signal(I2S), no external analog-to-digital converter (ADC) is needed, which can be useful for a future design handling multiple MEMS microphones.

2.1.2 Environmental sensor. The Bosch BME280 is a combined humidity, pressure and temperature sensor [6]. It transmits its sensor data through the I2C protocol. To read the sensor data at a frequency of 1 Hz, it consumes 3.6μA at 3.3V supply voltage on average. Further, it has the feature to tune the time between each measurement and to return in *sleep mode* in between. In this mode, the current consumption can be reduced to 0.1μA.

2.1.3 Microcontroller. The MAX78000 is an ultra low-power microcontroller focused on AI tasks [14]. Its convolutional neural network (CNN) accelerator makes it ideal for performing convolutional operations while maintaining a low energy consumption. As said before, it supports the I2C and I2S protocols, so it is well suited for time-series processing. It has an ARM Cortex-M4F, 128KB of RAM memory and 512KB of flash memory.

Table 1: Cost estimation of Aurora for producing a single protoype.

Component	Model	Quantity	Estimated price
Microcontroller	MAX78000FTHR	1	€ 31
3D printing	PLE filament	150 grams	€ 7
Battery	ICR18650 4400mAh 3.7V	1	€ 20
Environmental sensor	BME280	1	€ 13
Wires			€ 2
Total price			€ 75

2.1.4 Battery. The battery shown in Fig. 3 is a lithium-ion battery ICR18650 with a capacity of 4400 mAh at 3.7 V [16]. It could be charged with a solar panel [1] via a charger [2] to increase the battery lifetime of the board.

Figure 3: Lithium-ion battery with a capacity of 4400 mAh at 3.7 V.

2.2 Sensors

2.3 Box design

The design of the enclosure shown in Fig. 5 was made using a 3D printer. The top part of the enclosure is made with Polylactic Acid (PLA), and the other parts with nGen colorFabb [8]. These materials are waterproof and robust when sufficiently thick. The thickness of our components was of 2mm. The motivation behind an open design is firstly to not isolate the microphone from outside and limit the loss of sound level, and secondly to not encapsulate the environmental sensor. This is important for maintaining a high sensitivity of the humidity sensor. The waterproof property of the entire system is also ensured by making the rooftop sufficiently long and inclined, as depicted in Fig. 6.

2.4 Cost estimation

The total cost of the weather station can be found in the Table 1. This estimation reflects the cost to develop only a single prototype of Aurora. By economies of scale, the cost will be lower to produce multiple devices. The cost could be reduced even further by designing a proper Printed Circuit Board (PCB) with all necessary components on-board.

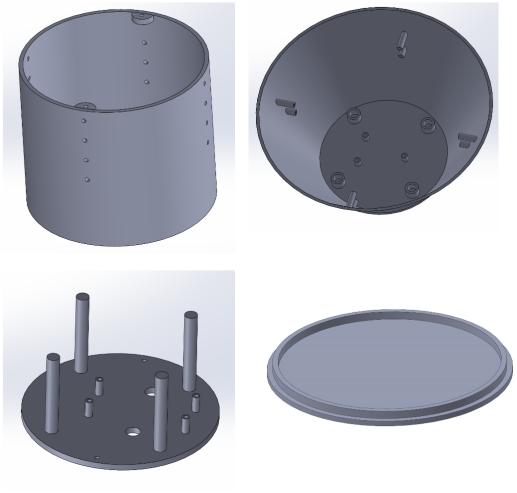

Figure 4: Design of 3D-printed enclosure.

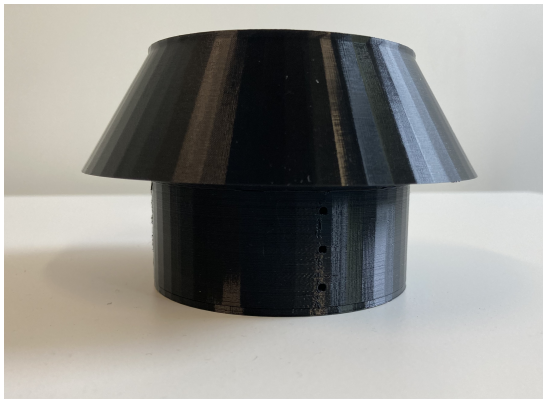

Figure 5: Aurora's final 3D-printed enclosure.

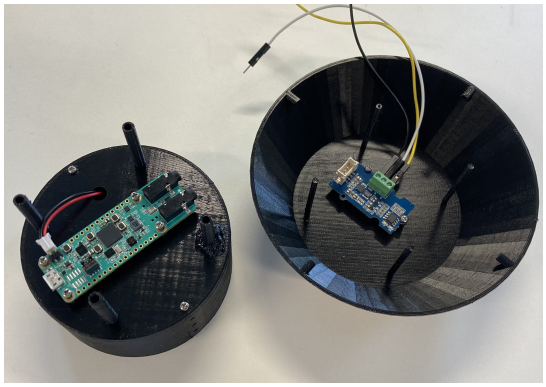

Figure 6: Enclosure inside: the MAX78000FTHR (left) and the Bosch BME280 sensor (right).

3 EMBEDDED APPLICATION

3.1 Real-time OS: μ111

The μ111 is a real-time OS for embedded systems developed by CSEM [9]. It provides a large panel of benefits. The first one is that it is designed for low-power systems supporting a variety of CPUs and SOCs. Secondly, it is composed of a collection of tools to build real-time applications. It has a low memory footprint and supports multi-core applications. Finally, it can run various machine learning (ML) algorithms, as it supports the Tensorflow Lite Micro library, as explained in the next subsection.

3.2 Tensorflow Lite Micro

The Tensorflow Lite Micro (TFLM) package provides a collection of tools to deploy efficiently an ML model, while being sufficiently small to fit into the memory of a microcontroller. For instance, it provides an efficient way to quantize the weights of the model by computing from a representative dataset an efficient dynamic range while not neglecting the accuracy of the model. Furthermore, it is quite straightforward to get from a Keras or Tensorflow model a Tensorflow lite micro model. The models deployed are quantized to take less memory.

4 DATASET

The dataset is acquired with the MAX78000 microphone and the environmental sensor, which estimates the temperature, humidity and pressure level. To get ground truth information about the wind and rain level, we utilize an Arduino-based weather station [19]. Powered by a micro:bit card, it records every two seconds the number of turns of the anemometer and the number of interrupts due to the tipping bucket rain gauge. The idea behind this weather station is to use it as the ground truth to label the data. We chose this type of weather station because most of the modern weather stations that we can find in the market, such as the DNT Weather Station Pro (which we used at the start of the project), have a granularity of measurements in the order of one minute. This make the correlation between the sound recording and the ground truth difficult, because of timeliness of wind and rain. For instance, it is possible to have no wind/rain within 30 seconds, and only at the end of the minute detect wind or rain.

Another major detail to take into account is how the tipping bucket rain gauge works. Inside of it, it has a small seesaw-like container that must be filled to tip and to send a signal that it has detected rain. However, when there is only a small amount of rain continuously falling, one can wait up to a few minutes before the container is filled and tip. And when looking only at the time graph of the rain interrupts, one cannot determine if the interrupts were caused by a constant rain of low level or by showers.

To tackle this issue, we have manually labelled the dataset by listening directly to the audio recording to detect when the rain has begun and stopped to fall, and use the number of interrupts within a period of time to infer the rain intensity. We have classified the rain intensity in three levels :

(1) 0 : no rain
(2) 1 : little rain, less than 1-2 mm per hour
(3) 2 : rain, more than 1-2 mm per hour

For the wind level however, we do not have to do any manual intervention and can use directly the number of interrupts that the anemometer triggers within a second, and then split the values into three classes :

(1) 0 : no wind, speed less than 6 km/h
(2) 1 : little wind, speed between 6 and 12 km/h
(3) 2 : wind, speed more than 12 km/h

Each entry of the dataset consists of the audio recording of duration of 1 second sampled at 16384 Hz, the humidity, temperature and pressure at this time, the relative direction of the wind, the rain level, the number of turns of the wind anemometer and the timestamp of the recording as identifier. The dataset is in the pickle format and can uncompressed into a Python Panda dataframe as follows:

```
import pandas as pd
df = pd.read_pickle(filepath)
print(df.columns)

['timestamp', 'humidity', 'pressure', 'temperature',
     'audio', 'wind_dir', 'wind_count', 'rain_count',
     'wind_x', 'wind_y']
```

The dataframe is generated using the version 1.4.3 of the Pandas library. Aurora was installed on the roof of the CSEM headquarters in Neuchâtel, Switzerland as shown in Fig. 7. Since it is placed not far from the road, it contains realistic sounds such as cars, buses, pedestrians and birds.

The dataset size contains around 165 hours of audio recordings in total.

5 ML MODEL

Multiple techniques exist to process audio signals for a given machine learning (ML) task. One well-known technique is to compute for each time window the corresponding Mel Frequency Cepstral Coefficients (MFCCs), which results in a two-dimensional input that can be processed by a neural network as an image. Nevertheless, in recent publications, feeding directly the raw signal of each time window to the ML model has shown better performance for audio-based tasks, leading to state-of-the-art results. As an example, the ESC-50 is as a labeled collection of 2000 environmental audio recordings [20]. It is mainly used as a benchmark for environmental audio classification methods. For this dataset, the best results were all obtained by processing directly the raw signal and letting the model learn internally an efficient embedding [7, 11]. However, most of these works use a lot of parameters in their model architectures consisting of complex layers such as transformers that could not easily be deployed on an embedded device. For this task, a recent paper presented a model called ACDNet [15] which performs very well even if only using convolution layers processing directly the raw signal.

Furthermore, to compute the MFCCs on an embedded device, depending on the board one should implement it from scratch and perform approximations in the algorithm to be memory efficient. By letting the ML model process directly the signal, we are sure that every loss of accuracy is due entirely to the model and thus can be spotted before porting the model.

Figure 7: Real-world dataset acquisition on the roof of the CSEM Headquarters in Neuchâtel, Switzerland.

Our model is inspired from one of the various models published by Wei Dai et al., processing directly the raw signal [10]. These models were also used to assess the rain level using microphones [4, 5]. In these publications, the authors have also proposed, still based on the same model, another model that can be deployed into an embedded device [17]. The given paper introduces several architectures with increasing complexity: M3, M5, M11, M18 and M34-res. Their global architecture consists of successive blocks made of a convolution layer followed by a max-pooling layer. The difference between those models is hence the number of such blocks.

The model proposed here aims to detect the rain and the wind level at the same time. It takes as input an audio recording of one half second sampled at 16384 Hz and classify it into the three classes for each tasks as defined in Section 4. For the last layer, the output of the dense layer is split into two parts, and a Softmax function is applied for each part to get a probability distribution for the rain in one side, and for the wind in the other side.

6 RESULTS

6.1 Demonstration

A video was made to demonstrate how the rain prediction of the model embedded in the MAX78000 performs in reality. For test purposes, the predictions were made one after the other, and it does not include the environmental measurements. The final application only performs one ML inference every minute and shows the environmental measurements at the same time.

Table 2: ML model architecture.

Layer	Output shape	Param #	Filter #	Kernel size	Stride/pool size
Conv1d	(None, 1639, 19)	1349	19	70	5
Max pooling	(None, 409, 19)				4
Conv1d	(None, 409, 96)	7392	96	4	1
Conv1d	(None, 409, 96)	27744	96	3	1
Conv1d	(None, 409, 96)	27744	96	3	1
Max pooling	(None, 102, 96)				4
Conv1d	(None, 102, 119)	34391	119	3	1
Max pooling	(None, 25, 119)				4
Global average	(None, 119)				
Dense	(None, 110)	13200	110		
Dense	(None, 100)	11100	100		
Dense	(None, 6)	606	6		

6.2 ML accuracy

To fine-tune the model, we have used the automatic hyperparameter optimization framework Optuna [3]. It proposes different methods to sample the hyperparameters such as CMA-ES [12] or GP-BO [18], each aiming to minimize a certain non-linear objective function. We define this function to be the accuracy of the model of interest over a test dataset. In addition, it provides pruning techniques to infer, during training, if it is relevant or not to continue the training by comparing the evolution of the accuracy over a validation set with other previous trials, and if not to stop the model and test another model with different parameters. Due to the imbalanced class, we use under-sampling techniques [13] and class re-weighting. By considering that we have at our disposal 128 KB for the RAM memory (cfr. Section 2.1.3), and taking into consideration that TFLM needs to allocate enough memory to store the result of each step of the model, one has to restrict the output shape size of each layer. It yields to the following model (cfr. Section 2).

The dataset is split in three subsets: the train set, the validation set and the test set. The model is trained during 40 epochs, and the weights of the epoch with the highest validation accuracy are at the end of the training process loaded into the model. We use a cross-entropy loss for each of the two outputs. The used optimizer is Adam with an initial learning rate of $1e^{-3}$, combined with a callback reducing by a factor 10 when the validation accuracy stagnates with a patience value of 3 (i.e. reduce the learning rate when it does not see improvements after three epochs), and a minimal learning rate of $5e^{-8}$.

When evaluating our model on the test set, we obtain the confusion matrices visualized in Fig. 10 and Fig. 9. As it appears, the model has no difficulty to classify samples with no rain. It performs less well on samples in which there is only a small amount of rain, and classify them sometimes has "no rain" or "rain". This behavior is confirmed when we study the second confusion matrix in Fig. 9 in which the samples "little rain" and "small rain" are merged together. Since the dataset is highly imbalanced, we use the balanced accuracy as the metric. One can see that the error is smaller. This is confirmed by the accuracy of the model: 77.50% in the three-class problem and 86.86% for the two-class problem.

We note that when we use the same model, but train it to only classify either rain or wind levels, but not both, we obtain a better

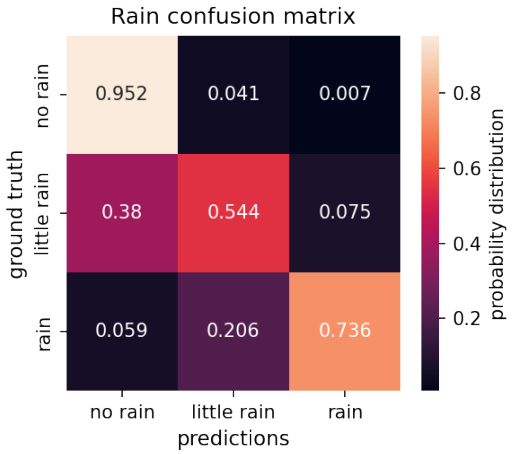

Figure 8: Three-class confusion matrix for the rain

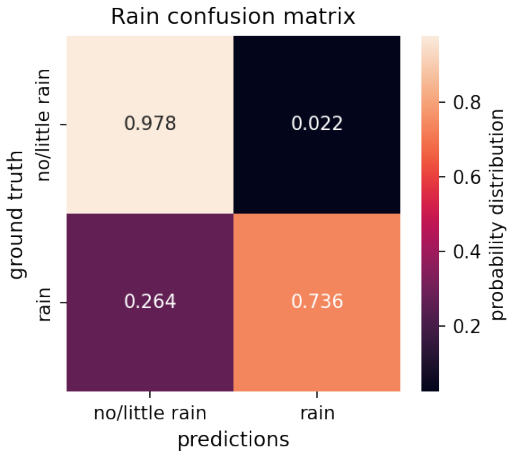

Figure 9: Two-class confusion matrix for the rain

Table 3: Aurora's memory footprint.

Component	Flash memory [KB]	RAM memory [KB]
Real-time OS	96.0	23.8
TFLM Library	220.2	3.0
ML model weights	150.5	-
ML model activations	-	86.5
Total	466.7	113.3

accuracy: 77.50% and 86.86% for the two and three-class respectively for the rain, 71.2% and 87.8% for the wind. We also observed that the accuracy increases by 3% to 5% when the audio length of the samples is one second instead of one half. However, it was not possible to deploy this model on the TinyML board due to the limited available RAM memory.

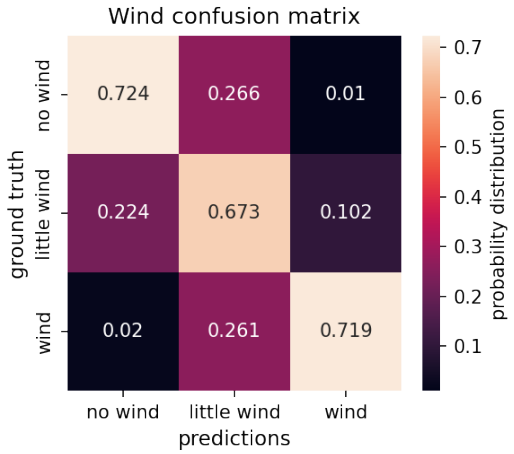

Figure 10: Three-class confusion matrix for the wind

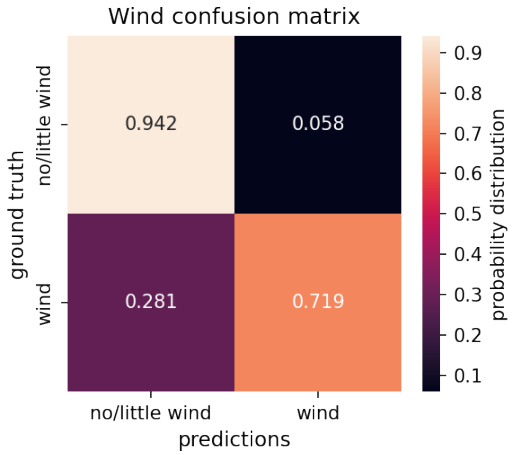

Figure 11: Two-class confusion matrix for the wind

6.3 Memory footprint

An overview of Aurora's memory footprint is presented in Table 3. Both the flash and RAM memory consumption is smaller than the available memory on the MAX78000. We note that the real-time OS currently contains all functionalities it offers, hence the rather large memory footprint. This can be reduced by developing a targeted implementation, containing only the necessary components of the OS for this application. With this in place, the flash memory footprint is expected to reduce to around 30 KB. This would also significantly lower the RAM footprint of the OS.

6.4 Energy consumption

We evaluated the energy consumption of the prototype using the current waveform analyzer Keysight CX3324A. In active mode, the current consumption of the board is ±12.5 mA at 3.7 V battery supply voltage. It includes the data extraction of the microphone and the environmental sensor, as well as the ML inference. However,

Table 4: Expected energy consumption for a single measurement, when designing a dedicated PCB. The components represented with the * are assumed to be active all the time.

Component	Current [mA]	Duration [s]	Energy [mAs]
MAX78000 in active mode	1.3	7.0	9.4
MAX78000 in sleep mode	$6.4e^{-1}$	$9.0e^{-1}$	$5.7e^{-1}$
SPH0645LM4H-B microphone	$6.0e^{-1}$	3.1	1.9
BME280 sensor	$3.6e^{-3}$	$1.6e^{-2}$	$5.7e^{-5}$
ISL9122A Buck-boost regulator*	$1.3e^{-3}$	-	-
Accessory components*	$10e^{-3}$	-	-
Total		7.9	11.8

the prototype contains other hardware components which are not needed for the application. Nevertheless, they consume a significant amount of energy.

In our future work, we would like design a dedicated Printed Circuit Board (PCB) that contains only the necessary components to minimize Aurora's energy consumption. Table 4 presents the expected energy consumption for one measurement. The reported current represents the current provided by the datasheet of the component's supplier. The reported duration represents the measured time by the firmware during which the respective component is active. These two values give the expected energy. The accessory components account for the passive elements not taken into account in this estimation. Together with the regulator, they are assumed to be activated all the time. Within one measurement, we put the MAX78000 partly in sleep mode while the microphone is being initialized. In between measurements, the MAX78000 is put in ultra low-power mode, which reduces its consumption to 33.3 μA.

Assuming we would perform one measurement per minute, the daily energy consumption would be 5.7 mAh. With a battery capacity of 4400 mAh, this would give Aurora a battery lifetime of around 770 days. We note that this is an idealized estimation, with the goal to give a rough idea of what could be achieved. It considers ideal conditions and neglects elements which could lower the performance, such as parasitic effects and battery degradation over time.

7 GOING FURTHER

To improve the performance of our wind and rain level measurements further and make it possible to differentiate among more level intensities, we aim to investigate sensor fusion techniques to combine the data from the environmental sensor and the microphone into one data stream to the ML model.

In addition to detecting the level of wind and rain, we would like to detect also the direction of wind. To achieve this, we would attach four microphones to the sides of the enclosure under the roof. These microphones would be connected via an I2S-multiplexer to our TinyML board.

To test more complex ML models, the usage of another microcontroller is considered, since the used microcontroller MAX78000 has only 128 KB of SRAM. Its advantages in terms of memory and low-power consumption come only when we use their developed framework providing CNN accelerator support instead of Tensorflow Lite Micro (TFLM). This restriction makes the development harder since we would need to use their API for this microcontroller, and we also loose portability when we want to use the embedded application in another device.

Further, to have a fully autonomous weather station, we want to install a solar panel for a cost around 20 euros [1], which could charge the battery and extend the lifetime of Aurora.

The final missing piece in our solution is the wireless communication towards the cloud. Until now, Aurora processes all data locally, but does not interact with the outside world. By allowing a remote connection with Aurora, we can assess the weather conditions remotely. This could be realized by using the low-power wide-area network LoRa [21]. This would further increase the impact of our weather station Aurora.

REFERENCES

[1] Adafruit. 2009. Small 6V 1W Solar Panel. https://www.adafruit.com/product/3809. (2009). [Online; accessed 03-October-2022].

[2] Adafruit. 2011. Adafruit Universal USB / DC / Solar Lithium Ion/Polymer charger - bq24074. https://www.adafruit.com/product/4755. (2011). [Online; accessed 03-October-2022].

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. (2019). DOI: https://doi.org/10.48550/ARXIV.1907.10902

[4] Roberta Avanzato and Francesco Beritelli. 2020. An Innovative Acoustic Rain Gauge Based on Convolutional Neural Networks. *Information* 11, 4 (2020). DOI: https://doi.org/10.3390/info11040183

[5] Roberta Avanzato, Francesco Beritelli, Francesco Di Franco, and Valerio Francesco Puglisi. 2019. A Convolutional Neural Networks Approach to Audio Classification for Rainfall Estimation. In *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 1. 285–289. DOI: https://doi.org/10.1109/IDAACS.2019.8924399

[6] Bosch. 2015. BME280, Combined humidity and pressure sensor. https://raw.githubusercontent.com/SeeedDocument/Grove-Barometer_Sensor-BME280/master/res/Grove-Barometer_Sensor-BME280-.pdf. (2015). [Online; accessed 03-October-2022].

[7] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2022. HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection. (2022). DOI: https://doi.org/10.48550/ARXIV.2202.00874

[8] colorFabb. 2015. nGen filament. https://c-3d.niceshops.com/upload/file/AmphoraAM3300-TDS.pdf. (2015). [Online; accessed 14-November-2022].

[9] CSEM. 2021. u111, LowPower RTOS+OS for secured IoT applications. https://www.csem.ch/pdf/159903?name=Micro_u111.pdf. (2021). [Online; accessed 03-October-2022].

[10] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. 2016. Very Deep Convolutional Neural Networks for Raw Waveforms. (2016). DOI: https://doi.org/10.48550/ARXIV.1610.00087

[11] Avi Gazneli, Gadi Zimerman, Tal Ridnik, Gilad Sharir, and Asaf Noy. 2022. End-to-End Audio Strikes Back: Boosting Augmentations Towards An Efficient Audio Classification Network. (2022). DOI: https://doi.org/10.48550/ARXIV.2204.11479

[12] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. DOI: https://doi.org/10.1162/106365601750190398

[13] imbalanced learn. 2019. Undersampling techniques. https://imbalanced-learn.org/stable/under_sampling.html. (2019). [Online; accessed 14-November-2022].

[14] Maxim Integrated. 2020. MAX78000, Artificial Intelligence Microcontroller with Ultra-Low-Power Convolutional Neural Network Accelerator. https://www.maximintegrated.com/en/products/microcontrollers/MAX78000.html. (2020). [Online; accessed 03-October-2022].

[15] Md Mohaimenuzzaman, Christoph Bergmeir, Ian Thomas West, and Bernd Meyer. 2021. Environmental Sound Classification on the Edge: A Pipeline for Deep Acoustic Networks on Extremely Resource-Constrained Devices. (2021). DOI: https://doi.org/10.48550/ARXIV.2103.03483

[16] PKCELL. 2014. Polymer Li-ion Battery Technology Specification. https://cdn-shop.adafruit.com/product-files/354/C449_-_ICR18650_4400mAh_3.7V_with_PCM_20140728_APPROVED_8.18.pdf. (2014). [Online; accessed 03-October-2022].

[17] Michele Russo, Valerio Francesco Puglisi, Roberta Avanzato, and Francesco Beritelli. 2021. A CNN-based Audio Sensor for Rainfall Estimation: Implementation on Embedded Board. In *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 2. 911–915. DOI: https://doi.org/10.1109/IDAACS53288.2021.9660891

[18] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. DOI: https://doi.org/10.1109/JPROC.2015.2494218

[19] SparkFun. 2019. SparkFun micro:climate kit. https://www.sparkfun.com/products/16274. (2019). [Online; accessed 14-November-2022].

[20] Ivo Trowitzsch, Jalil Taghia, Youssef Kashef, and Klaus Obermayer. 2019. The NIGENS General Sound Events Database. (2019). DOI: https://doi.org/10.48550/ARXIV.1902.08314

[21] Wikipedia. 2019. LoRa. https://en.wikipedia.org/wiki/LoRa. (2019). [Online; accessed 14-November-2022].