# Tiny Machine Learning for Damage Classification in Concrete Using Acoustic Emission Signals

Veysi Adın, Yuxuan Zhang, Bengt Oelmann and Sebastian Bader
*Department of Electronics Design*
*Mid Sweden University*
Sundsvall, Sweden
veysi.adin@miun.se, yuxuan.zhang@miun.se, bengt.oelmann@miun.se, sebastian.bader@miun.se

***Abstract*—Acoustic emission (AE) is a widely used non-destructive test method in structural health monitoring applications to identify the damage type in the material. Usually, the analysis of the AE signal is done by using traditional parameter-based methods. Recently, machine learning methods showed promising results for the analysis of AE signals. However, these machine learning models are complex, slow, and consume significant amounts of energy. To address these limitations and to explore the trade-off between model complexity and the classification accuracy, this paper presents a lightweight artificial neural network model to classify damage types in concrete material using raw acoustic emission signals. The model consists of one hidden layer with four neurons and is trained on a public acoustic emission signal dataset. The created model is deployed to several microcontrollers and the performance of the model is evaluated and compared with a state-of-the-art machine learning model. The model achieves 98.4% accuracy on the test data with only 4019 parameters. In terms of evaluation metrics, the proposed tiny machine learning model outperforms previously proposed models 10 to 1000 times. The proposed model thus enables machine learning in real-time structural health monitoring applications.**

***Index Terms*—TinyML, damage classification, acoustic emission, structural-health-monitoring, embedded systems, IoT, machine learning**

## I. Introduction

Acoustic emission (AE) signals are well-known characteristic signals that are emitted when a material undergoes a deformation [1]. Using the AE signals, researchers were able to identify [2], localize [3], and characterize [4] the type of deformation that the material undergoes without causing external damage to the material. This method is thus a non-destructive testing method for the material. AE signals are captured with AE sensors, most often containing a piezoelectric element, as shown in Fig. 1. The mechanical energy of the elastic wave caused by a deformation is transformed into an electrical signal by the piezoelectric element inside the sensor [5]. These sensors can be considered as stethoscopes for identifying the source of damage in the material, which is the most essential part of the AE method. The AE method is vital for structural health monitoring (SHM) and is being used for a wide variety of materials and critical structures such as concrete [6], composites [7], bridges [8], nuclear power plants [9], metals [10], and pipelines [11].

An AE signal has several important parameters that are being used in the estimation of damage types, including rise time, amplitude, average frequency, peak frequency, signal strength, central frequency, count, energy, absolute energy, and duration [1]. As depicted in Fig. 2, several of the parameters can be extracted from the AE signal. Conventional methods for identifying damage types include the analysis of these parameters and the waveform of the signal. Examples of these analysis methods use simplified Green's functions for moment tensor analysis (SiGMA) [12], b-value analysis [13], Hilbert-Huang transform [14], wavelet transform, and Shannon entropy [15]. The advantage of parameter-based analysis is that using the same parameters, damage type and source location can be identified. However, parameter-based analysis has reproducibility and generalization problems due to the different material types and the noisy AE signal [16]. Additionally, the analysis requires significant effort from material experts [17].

More recently, machine learning emerged as a solution to overcome the limitations of parameter and waveform analysis methods. For example, in [18] a variant of the k-means clustering method is used for the damage classification of AE signals.

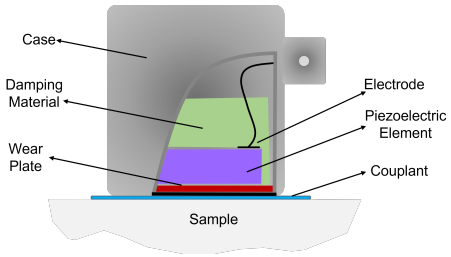


Fig. 1. Internal structure of an example AE sensor with key components

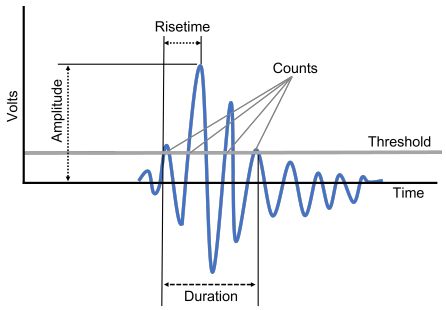


Fig. 2. Typical parameters that can be extracted from an AE signal

In [19], support vector machines (SVM) are used for the crack classification in concrete material. Researchers applied most of the machine learning methods available for damage classification using AE signals. Most of the machine learning models available in the literature using AE signals, including multi-layer perceptron (MLP) network [20], long short-term memory (LSTM) based network [21], gated recurrent unit (GRU) network [22], convolutional neural networks (CNN) [23], have achieved accuracies higher than 95% and the details of the machine learning based methods applied in AE testing are reported in [16].

While the main concern for the aforementioned models is to achieve high accuracy for the damage classification, the model size, inference time, number of parameters, energy per inference, and memory usage remains an issue for the deployment of the models to embedded systems for real-time SHM applications. To the best of our knowledge, damage classification using AE signals with machine learning methods has not been implemented on embedded systems before. Therefore, in this paper, we focus on the development and implementation of a lightweight machine learning model that is deployable to a wide range of microcontrollers, to identify damage type in concrete using raw AE signals. The implemented model is deployed to six different widely available microcontrollers having different CPU, flash size, and RAM size. To explore the trade-off between model complexity and other evaluation metrics, the implemented model is evaluated and compared with a lightweight CNN model described in [24] in terms of accuracy, model size, inference time, number of parameters, memory and flash usage, as well as energy per inference.

Section II of this paper describes the AE dataset used for the training and validation and details the development, implementation, and deployment of the proposed model. Section III presents the results of the model evaluation and the comparison of the proposed model with the state-of-the-art. Finally, the conclusion and future works are given in Section IV.

## II. Method

### A. Dataset

The dataset utilized in this paper was originally described in [21] and is openly accessible. The AE signals in this dataset are acquired by a multi-triggered acquisition system, in which five asymmetrically arranged AE sensors are attached to the concrete block under test. Fig. 3 depicts the experimental setup for the acquisition of the AE signals. The AE data acquisition system consists of a hydraulic press, a $15\,\mathrm{cm}^3$ concrete block without any reinforcement which is prepared based on the standard in [25], five AE sensors (piezoelectric transducers), logic flat amplifiers, and data acquisition boards with 12-bit resolution and $5\,\mathrm{MHz}$ sampling frequency. More information about the setup, can be found in [21].

The dataset contains 15000 raw acoustic emission signals of $2\,\mathrm{ms}$ duration. Each of the signals in the dataset thus consists of 10000 data points. The AE signals were captured by five identical sensors, but only the signal with highest signal-to-noise ratio (SNR) is represented in the dataset. Each signal is down-sampled to 1000 datapoints to reduce processing time. The signals are labeled based on the damage types in concrete, namely, tensile, shear, and mixed damage type. The dataset is balanced with respect to the three damage classes, which means that it contains 5000 samples for each damage class. Examples of AE signals in the dataset for each damage type are shown in Fig. 4.

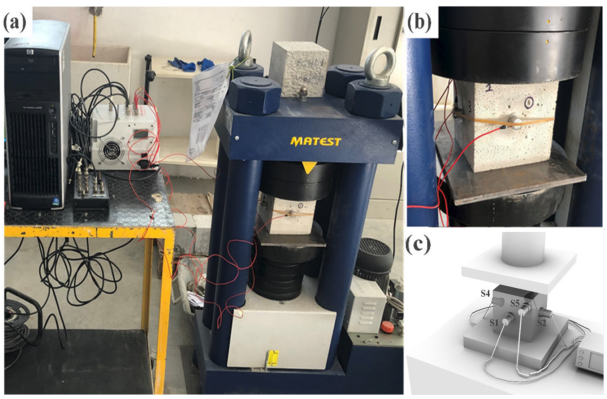


Fig. 3. (a) AE signal data acquisition experimental setup. (b) Detail of the sample under test with AE sensors. (c) A sketch of the AE signal acquisition setup [21]

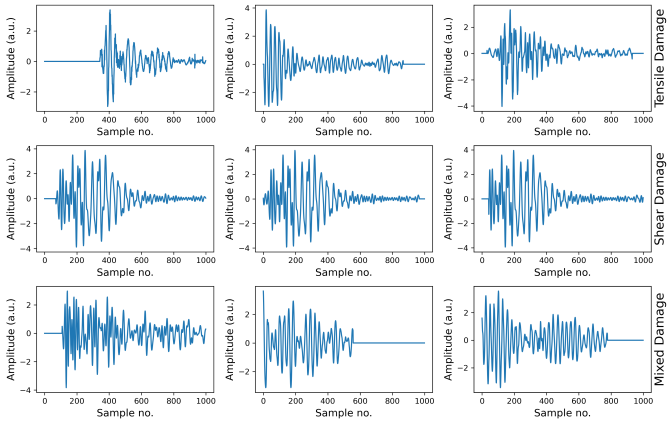


Fig. 4. Examples of AE signals for each damage type in the dataset

The dataset is split into training, validation and testing datasets with a ratio of 70%, 15%, 15%, resulting in 10500, 2250, and 2250 samples, respectively.

### B. Artificial Neural Network Model Design

Artificial neural networks (ANN) are a type of machine learning inspired by biological neurons. Each neuron is connected to another neuron creating a layered network, and each neuron is a node in the network. As shown in Fig. 5(a), ANNs usually consist of an input layer, one or more hidden layers, and an output layer. Except for the input layer, each ANN layer receives inputs from the previous layer, computes the weighted total of the inputs, incorporates a bias, and feeds the output to an activation function such as a rectified linear unit (ReLU), sigmoid, or exponential linear unit (ELU). The

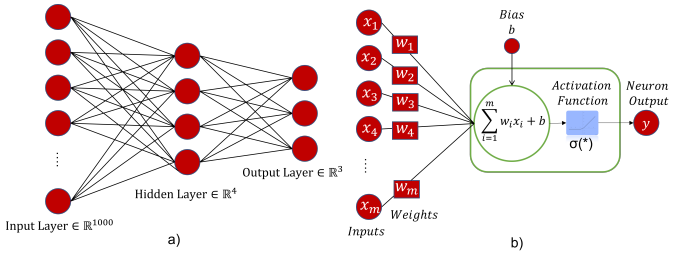


Fig. 5. a) Artificial neural network model structure b) Perceptron block

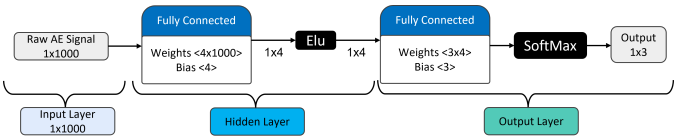


Fig. 6. Developed ANN model structure

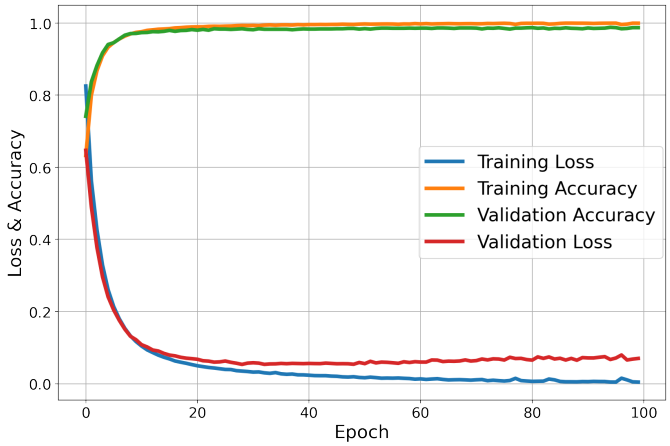


Fig. 7. Training and validation results of the proposed ANN model developed and implemented in Tensorflow

computation of each node is visualized in Fig. 5(b), and its transfer function is given as

$$f(x) = \sigma\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{1}$$

where $x_i$ are the inputs of the node, $w_i$ are the weights, $b$ is the bias, $m$ is the number of inputs and finally $\sigma$ is the nonlinear activation function.

The proposed ANN model is depicted in Fig. 6. It consists of an input layer with an input size of 1x1000, one hidden layer consisting of four neurons with ELU [26] activation function, and an output layer consisting of three neurons for the three different damage types with a respective softmax activation function for each output node. In comparison to units with different activation functions, ELUs have better learning qualities. ELUs, in contrast to ReLUs, have negative values, allowing them to achieve batch normalization-like behavior by bringing mean unit activations closer to zero, but with a reduced computational cost [26]. The mathematical representation of the ELU activation function is given in (2), where $\alpha$ is the scale for the negative factor and in our implementation, the default $\alpha$ value of one is used.

$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x \leq 0 \\ x & \text{for} \quad x > 0 \end{cases} \tag{2}$$

### C. Training and Validation

The model is trained using the TensorFlow library due to the convenience of deployment to different microcontrollers with the TensorFlow Lite Micro (TFLM) framework [27]. The loss function for the model is sparse categorical cross-entropy loss, and the model is trained for 100 epochs with the Adaptive Moment Estimation (Adam) optimizer. The optimum learning rate and batch size are determined using the Weights & Biases hyperparameter sweep interface [28], resulting in a batch size of 32 and a learning rate of 0.0005, respectively. For reproducibility purposes, a fixed random seed is used, so that the network can be retrained with the same results. The code for training and deployment is open-source and was made available in [29]. The training and validation results are shown in Fig. 7. The model achieves 99.94% training accuracy, 0.00045 training loss, 98.71% validation accuracy, and 0.0702 validation loss.

### D. Model Optimizations and Model Deployment

Machine learning models can be deployed to embedded systems using the open-source TensorFlow Lite Micro (TFLM) ML inference framework [27]. TensorFlow Lite (TFL) [30] allows machine learning models to be optimized for memory footprint, model size, inference time, and power consumption, using various methods such as quantization, pruning, and weight clustering. Many machine learning models are optimized using TFL. After the optimization, usually model size is reduced by a factor of 1.5–4, and inference efficiency in terms of latency, memory footprint, and power consumption is improved by a factor of 1.5–4 [27].

In our implementation, both the original and a quantized version of the developed model are deployed to six common microcontrollers. Key properties of each microcontroller are shown in Table I. TFLM does currently not support batch normalization and dropout layers, which are being used in the CNN model proposed in [24]. For comparison reasons, this model is modified for TFLM compatibility with negligible change in accuracy. Moreover, a quantized version of the model is generated and both models are deployed to the same microcontrollers. The deployed models are compared in terms of training accuracy, training loss, validation accuracy, validation loss, model size, average inference time, energy per inference, and memory footprint. The workflow starting from the design of the ANN model to the performance measurement of the deployed model is shown in Fig. 8.

Initially models are compared in terms of test accuracy and the number of parameters. Afterwards, the Machine Learning Toolkit (MLTK) library [31] is used for the comparison of models converted to TFL format. The MLTK library profiles

TABLE I
PROPERTIES OF THE MICROCONTROLLERS USED FOR DEPLOYMENT

| MCU / Hardware | ESP32-S | RP2040 | nRF52840 | STM32 F401xD | STM32 L432KC | SAMD21G18 |
|---|---|---|---|---|---|---|
| **Test Board** | Node MCU V1.3 ESP32-S | Raspberry Pi Pico | Arduino Nano BLE Sense 33 Lite | Nucleo F401RE | Nucleo L432 | Seeduino Xiao |
| **Core Processor** | Tensilica Xtensa | ARM Cortex M0+ | ARM Cortex M4 | ARM Cortex M4 | ARM Cortex M4 | ARM Cortex M0+ |
| **CPU Frequency** | 160 MHz | 133 MHz | 64MHz | 84MHz | 80 MHz | 48MHz |
| **SRAM** | 520KB | 256 KB | 256KB | 96KB | 64KB | 32KB |
| **Flash** | 4MB | 2MB QSPI | 1MB | 512KB | 256KB | 256KB |
| **FPU** | ✓ | X | ✓ | ✓ | ✓ | X |
| **Current Consumption per MHz** | 125µA* | 75µA | 52µA* | 146µA | 84µA | 132µA |
| **Voltage** | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V |
| **IDE / Framework** | Espressif IDF, C/C++ | Pico C/C++ SDK | Arduino IDE | Arduino IDE | Arduino IDE | Arduino IDE |

*Bluetooth and Wi-Fi are disabled

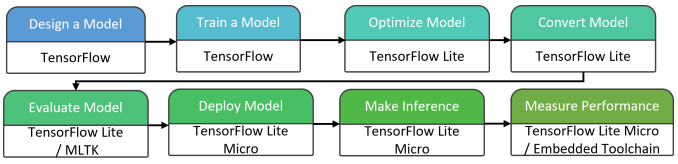


Fig. 8. TinyML model deployment workflow

the TFL models and provides hardware-independent information about the model's RAM usage and flash usage. Moreover, it provides the total number of operations, the CPU cycle counts, and estimates the energy for an inference on a ARM Cortex-M33 core [31]. Since the MLTK profiling results are based on one specific microcontroller core, as a final step of the performance evaluation, the models are deployed to the different microcontroller boards to explore the impact of different microcontrollers on the evaluation metrics.

## III. RESULTS AND DISCUSSION

The developed ANN model is tested on the test dataset, and the result of the test is depicted in the confusion matrix in Fig. 9. The model correctly predicted the damage classes for 2214 out of the 2250 test signals, achieving an average test accuracy of 98.4% as shown in Table II.

The developed ANN model is compared with the model in [24] in terms of the number of parameters, validation and test results, and the result of the initial comparison is shown in Table III. Although our model has 1.3% lower test accuracy, our model has ∼5x smaller number of parameters with 4,019 parameters while the lightweight model in [24] has 20,243 parameters.

### A. MLTK Profiling Results

After the initial comparison, the original models are converted to TFL format and profiled using the MLTK library.

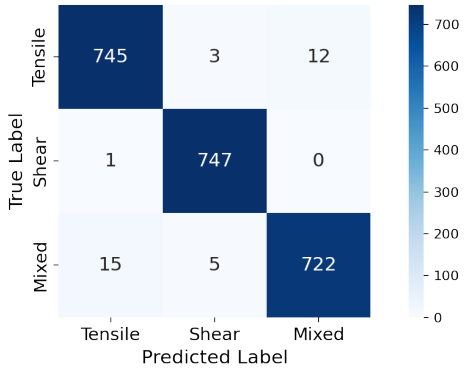


Fig. 9. Confusion matrix of the model evaluation on the test dataset

TABLE II
TRUTH TABLE, PRECISION, RECALL AND F1 SCORE

| | TP | FN | FP | Precision | Recall | F1 Score |
|---|---|---|---|---|---|---|
| Tensile | 745 | 15 | 16 | 0.9790 | 0.9803 | 0.9796 |
| Shear | 747 | 1 | 8 | 0.9894 | 0.9987 | 0.9940 |
| Mixed | 722 | 20 | 12 | 0.9837 | 0.9730 | 0.9783 |
| Average | - | - | - | - | - | 0.9840 |

Then the models are quantized using TFL's post-training quantization method [32] to convert model weights from 32-bit float to 8-bit integer. Quantization improves latency, model size, and processing power with little degradation in model accuracy. The quantized models are profiled using the MLTK library as well, and the result of the profiling for all four models (original and quantized) is shown in Table IV. According to the result of the comparison, post-training quantization decreased the model size by 3.5x for the CNN model [24], while our model size is decreased by 2.6x. Our model size is 12x smaller than the CNN model, while the quantized version is 9x smaller than the quantized CNN model. The total number of operations required for an inference is 1650x smaller for our model. RAM usage for our model is 15.7x smaller than the CNN model, while the RAM usage for the quantized version of our model is 5.9x smaller than the quantized CNN model. The estimated energy consumption per inference is ∼20,000x smaller for our model.

### B. Deployment Results

The models are finally deployed to microcontroller boards, and average inference times are acquired for 1000 samples per microcontroller board. Due to its model size, it was not possible to deploy the CNN model [24] to all microcontrollers. Models are stored in flash memory and the CNN model has a

TABLE III
MODEL TRAINING, VALIDATION AND TEST RESULTS COMPARISON

| Model | Parameters | Val Acc | Val Loss | Test Acc |
|---|---|---|---|---|
| Model [24] | 20,243 | **99.70** | **0.0155** | **99.70** |
| **Our Model** | **4,019** | 98.71 | 0.0702 | 98.40 |

TABLE IV
MODEL COMPARISON USING MLTK LIBRARY

| Model | Model Size (kB) | Total Ops | RAM Usage (kB) | CPU Cycle Count[1] | Energy per Inference (J)[1] |
|---|---|---|---|---|---|
| Model [24] | 219.9 | 13.2M | 146.8 | 39.1M | 2.20m |
| Model [24] Quantized | 62.7 | 13.204M | 39.1 | 39.11M | 2.21m |
| **Our Model** | 18.2 | **8000** | 9.3 | **19.8k** | **109.20n** |
| **Our Model Quantized** | **7.0** | 12,100 | **6.6** | 71.2k | 1.30$\mu$ |

[1]Estimations are based on ARM Cortex-M33

size of 219.9 kB and needs 146.8 kB of RAM for the dynamic variables. Therefore, it was only possible to deploy the CNN model to the nRF52840 and RP2040. Although the ESP32-S has sufficient flash and RAM sizes, it was not possible to deploy the CNN model due to the additional memory required by FreeRTOS. The details of the inference times per board is depicted in logarithmic scale in Fig. 10. The measured inference times show that our model is 272–1000x faster than the CNN model, and that the quantized version of the model is 76–220x faster than the quantized CNN model. For our model, among the deployed microcontrollers, the ESP32-S has the fastest average inference time with 400±3.18 µs (average inference time±standard deviation), while the SAM21G18 has the slowest inference time with 32438±2.77 µs per inference. For the CNN model the average inference time is 1.13 ± 0.0017s on NRF52840 and 3.23 ± 0.018s on RP2040, while for the quantized CNN model ESP32-S has the fastest average inference time with 172.1±0.72 ms, and NRF52840 has the slowest inference time with 626.22±1.29 ms. The reason for faster inference of our model is that it has a lower number of parameters, lower number of layers and lower number of operations per inference. Additionally, the quantized ANN model is slower on microcontrollers with floating point unit (FPU), because the quantization adds two layers to the model for quantization and dequantization, respectively. The addition of those layers increases the total number of operations per inference by 150% for the ANN model, thereby increasing the inference time. On the microcontrollers without FPU the quantized ANN model has a faster inference time, due to the fact that software emulation of FPU is avoided. However for the CNN model, quantization improves the inference time by 2–6x. This is due to the fact that the CNN model has 13.2M operation per inference and quantization and dequantization layers only adds 4000 extra operations, with faster 8-bit integer access.

Fast inference time is beneficial for the energy consumption as shown in the energy consumption per inference plot in Fig. 11. The plot shows estimated energy consumption per inference for each microcontroller. The energy consumption is estimated by multiplying the average inference time with the current consumption and the operating voltage according to the datasheet of each microcontroller. For our model, among the six microcontrollers, the NRF52840 has the lowest energy consumption with 11.8 µJ per inference, and the SAMD21G18 has the highest energy consumption with 675.5 µJ per inference.

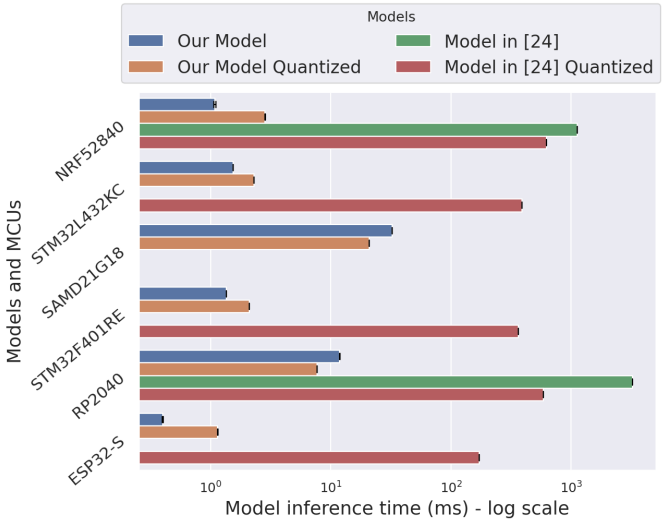


Fig. 10. Inference times of the models deployed on microcontrollers

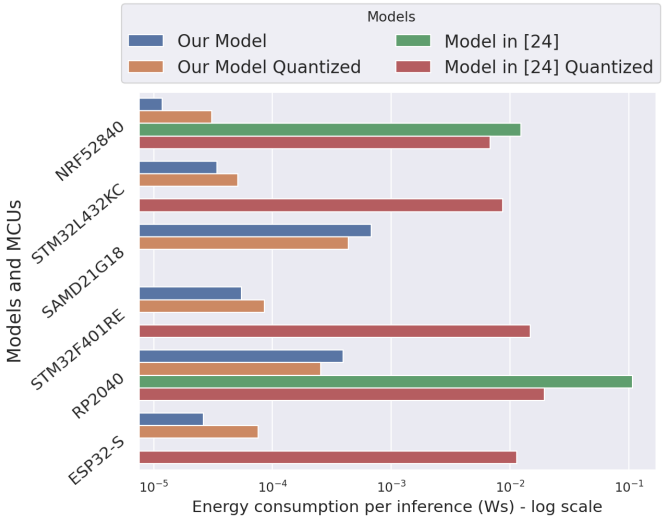


Fig. 11. Energy consumption per inference of the models deployed on microcontrollers

According to the energy consumption per inference, our model is 272–1000x more energy efficient than the CNN model, and the quantized version of the model is 76–220x more energy efficient than the quantized CNN model.

## IV. CONCLUSIONS AND FUTURE WORK

In this work, we studied the trade-off between model complexity and accuracy for real-time damage classification

on embedded systems. We developed a tiny ANN model for damage classification in concrete materials using a public AE signal dataset. The model was implemented, tested, and deployed to six different microcontrollers for performance analysis in terms of memory footprint, model size, inference time, and energy consumption. The model is compared with a state-of-the-art, lightweight CNN model in terms of relevant performance metrics. Moreover, the compared models are quantized, and the same comparison is done using quantized models. The proposed ANN model achieves an accuracy of 98.40% on the test set with a test loss of 0.0902. We showed that among the deployed microcontrollers ESP32-S has the fastest average inference time, while the NRF52840 has the lowest energy consumption per inference. Thus, simplifying microcontroller selection for applications requiring fast inference or low energy consumption. Although our model has a slightly lower (-1.3%) classification accuracy than the lightweight model previously proposed in [24], we demonstrate that a 12x improvement in the model size, 1000x improvements in the inference time, up to 1000x improvements in the energy consumption and up to 16x improvements in the memory footprint is possible. Therefore, these improvements in the model enable deployment to a wide range of embedded microcontrollers that can be used in real-time structural health monitoring applications.

## REFERENCES


[1] S. Mindess, "Acoustic emission methods," in *Handbook on nondestructive testing of concrete*, 2nd ed., V. Mohan Malhotra and N. J. Carino, Eds. ASTM International, Jul. 2004, ch. 16, pp. 348–362.

[2] S. Huguet, N. Godin, R. Gaertner, L. Salmon, and D. Villard, "Use of acoustic emission to identify damage modes in glass fibre reinforced polyester," *Composites Science and Technology*, vol. 62, no. 10, pp. 1433–1444, 2002.

[3] M. G. Baxter, R. Pullin, K. M. Holford, and S. L. Evans, "Delta t source location for acoustic emission," *Mechanical Systems and Signal Processing*, vol. 21, no. 3, pp. 1512–1520, 2007.

[4] A. Marec, J.-H. Thomas, and R. El Guerjouma, "Damage characterization of polymer-based composite materials: Multivariable analysis and wavelet transform for clustering acoustic emission data," *Mechanical Systems and Signal Processing*, vol. 22, no. 6, pp. 1441–1464, 2008, special Issue: Mechatronics.

[5] A. Behnia, H. K. Chai, and T. Shiotani, "Advanced structural health monitoring of concrete structures with the aid of acoustic emission," *Construction and Building Materials*, vol. 65, pp. 282–302, 2014.

[6] J. Geng, Q. Sun, Y. Zhang, L. Cao, and W. Zhang, "Studying the dynamic damage failure of concrete based on acoustic emission," *Construction and Building Materials*, vol. 149, pp. 9–16, 2017.

[7] E. Maillet, C. Baker, G. N. Morscher, V. V. Pujar, and J. R. Lemanski, "Feasibility and limitations of damage identification in composite materials using acoustic emission," *Composites Part A: Applied Science and Manufacturing*, vol. 75, pp. 77–83, 2015.

[8] A. Nair and C. Cai, "Acoustic emission monitoring of bridges: Review and case studies," *Engineering Structures*, vol. 32, no. 6, pp. 1704–1714, 2010.

[9] S. Mohd, K. M. Holford, and R. Pullin, "Continuous wavelet transform analysis and modal location analysis acoustic emission source location for nuclear piping crack growth monitoring," *AIP Conference Proceedings*, vol. 1584, no. 1, pp. 61–68, 2014.

[10] J. P. McCrory, A. Vinogradov, M. R. Pearson, R. Pullin, and K. M. Holford, *Acoustic Emission Monitoring of Metals*. Springer International Publishing, 2022, pp. 529–565.

[11] J. Lim, "Underground pipeline leak detection using acoustic emission and crest factor technique," in *Advances in Acoustic Emission Technology*, G. Shen, Z. Wu, and J. Zhang, Eds. New York, NY: Springer New York, 2015, pp. 445–450.

[12] M. Shigeishi and M. Ohtsu, "Acoustic emission moment tensor analysis: development for crack identification in concrete materials," *Construction and Building Materials*, vol. 15, no. 5, pp. 311–319, 2001, fracture Mechanics and Acoustic Emmission.

[13] I. S. Colombo, I. G. Main, and M. C. Forde, "Assessing damage of reinforced concrete beam using "b-value" analysis of acoustic emission signals," *Journal of Materials in Civil Engineering*, vol. 15, no. 3, pp. 280–286, 2003.

[14] S. E. Hamdi, A. Le Duff, L. Simon, G. Plantier, A. Sourice, and M. Feuilloy, "Acoustic emission pattern recognition approach based on hilbert–huang transform for structural health monitoring in polymer-composite materials," *Applied Acoustics*, vol. 74, no. 5, pp. 746–757, 2013.

[15] X. Zhang, N. Feng, Y. Wang, and Y. Shen, "Acoustic emission detection of rail defect based on wavelet transform and shannon entropy," *Journal of Sound and Vibration*, vol. 339, pp. 419–432, 2015.

[16] G. Ciaburro and G. Iannace, "Machine-learning-based methods for acoustic emission testing: A review," *Applied Sciences*, vol. 12, no. 20, 2022.

[17] V. Tra, J. Y. Kim, I. Jeong, and J. M. Kim, "An acoustic emission technique for crack modes classification in concrete structures," *Sustainability (Switzerland)*, vol. 12, 9 2020.

[18] E. Pomponi and A. Vinogradov, "A real-time approach to acoustic emission clustering," *Mechanical Systems and Signal Processing*, vol. 40, no. 2, pp. 791–804, 2013.

[19] A. K. Das, D. Suthar, and C. K. Leung, "Machine learning based crack mode classification from unlabeled acoustic emission waveform features," *Cement and Concrete Research*, vol. 121, pp. 42–57, 2019.

[20] O. Inderyas, S. Tayfur, T. Arslan, and N. Alver, "Deep learning-based damage estimation in concrete structures using acoustic emission data," *The International Symposium on Nondestructive Testing in Civil Engineering (NDT-CE 2022)*, 8 2022.

[21] G. Siracusano, F. Garescì, G. Finocchio, R. Tomasello, F. Lamonaca, C. Scuro, M. Carpentieri, M. Chiappini, and A. L. Corte, "Automatic crack classification by exploiting statistical event descriptors for deep learning," *Applied Sciences (Switzerland)*, vol. 11, 12 2021.

[22] X. Li, J. Li, Y. Qu, and D. He, "Gear pitting fault diagnosis using integrated cnn and gru network with both vibration and acoustic emission signals," *Applied Sciences (Switzerland)*, vol. 9, 2 2019.

[23] F. Guo, W. Li, P. Jiang, F. Chen, and Y. Liu, "Deep learning approach for damage classification based on acoustic emission data in composite materials," *Materials*, vol. 15, no. 12, 2022.

[24] Y. Zhang, S. Bader, and B. Oelmann, "A lightweight convolutional neural network model for concrete damage classification using acoustic emissions," in *2022 IEEE Sensors Applications Symposium (SAS)*, 2022, pp. 1–6.

[25] *Testing hardened concrete - Part 3: Compressive strength of test specimens*, EUROPEAN STANDARDS Std. DIN EN 12 390-3, 2019.

[26] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[27] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang, P. Warden, and R. Rhodes, "Tensorflow lite micro: Embedded machine learning for tinyml systems," in *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica, Eds., vol. 3, 2021, pp. 800–811.

[28] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[29] V. ADIN, "TinyML Model for Damage Classification in Concrete Using AE Signals," https://github.com/veysiadn/acustic-emission-models, 2022.

[30] Tensorflow, "Tensorflow lite — ml for mobile and edge devices," https://www.tensorflow.org/lite/, (Accessed on 10/31/2022).

[31] Silicon Labs, "Silicon labs machine learning toolkit (mltk) — mltk 0.12.0 documentation," https://siliconlabs.github.io/mltk/index.html, (Accessed on 10/31/2022).

[32] TensorFlow, "Post-training quantization," https://www.tensorflow.org/model_optimization/guide/quantization/post_training, (Accessed on 11/02/2022).