A low-power embedded system for fire monitoring and detection using a multilayer perceptron

Alexios Papaioannou
Information Technologies Institute
Thessaloniki, Greece
alexiopa@iti.gr

Panagiotis Verikios
Information Technologies Institute
Thessaloniki, Greece
panos_verikios@iti.gr

Charalampos S Kouzinopoulos
Information Technologies Institute
Thessaloniki, Greece
kouzinopoulos@iti.gr

Dimosthenis Ioannidis
Information Technologies Institute
Thessaloniki, Greece
djoannid@iti.gr

Dimitrios Tzovaras
Information Technologies Institute
Thessaloniki, Greece
Dimitrios.Tzovaras@iti.gr

***Abstract*—Fire monitoring and detection systems can evaluate data from environmental or image sensors in order to predict occurrences of fire. It is a complex procedure that requires a significant amount of energy as input data is usually acquired from multiple sensors and the algorithms generally have an increased complexity. This paper introduces a low-power fire monitoring and detection system that utilizes data from two environmental sensors. As a predictive algorithm for fire occurrences, it uses a multilayer perceptron (MLP) with a combination of different optimizations, developing a model with low memory requirements and high-accuracy predictions. The accuracy of the proposed system was verified using a dataset created by the environmental sensors for fire incidents and its performance was compared to existing approaches. An evaluation of the proposed system's power consumption and memory requirements is also presented.**

***Index Terms*—Fire detection and monitoring, Embedded systems, Low-power, MLP**

I. Introduction

Fire is a procedure of burning combustible materials. It is a chemical reaction that, when combined with oxygen, releases heat and flames and can often cause significant damages in personal property and casualties of human lives. Fire monitoring and detection systems are significant features of surveillance systems and can be used to detect fire in early stages in order to avoid its spread.

Two main methods of fire detection have been proposed in the literature. The first is based on environmental sensors such as temperature, humidity, smoke and gas [1], [2], [3] with the main goal being to detect fire incidents at an early stage, without memory or power consumption constraints. The second is based on a combination of image sensing and machine learning algorithms for image processing [4], [2]. Such a system can detect a fire from direct flames but only after the fire has spread.

Existing systems [1], [5] commonly use one or more sensors for fire monitoring or detection and usually without any memory or power constraints as the systems that execute the models have enough system resources and continuous power supply. Also, algorithms used for fire monitoring are complex and generally have increased memory requirements. Thus, such systems cannot be implemented on embedded systems with limited system resources. To fill this research gap, a low-power embedded system for fire monitoring and detection is proposed in this paper, using an efficient Neural Network. The low-power consumption and memory footprint of the proposed system, as well as its high-accuracy predictions, make it an innovative solution for integration into an embedded system.

The rest of the paper is structured as follows: Section II details previous research on fire monitoring and detection systems. In section III, the architecture of the proposed system is presented, along with different optimization techniques. In section IV, experimental results on energy consumption, memory requirements and the accuracy of the proposed model are presented with a comparison to existing approaches. Finally, section V presents the conclusions of this research and discusses future work.

II. Related Work

In the past few years, a growing number of articles on fire monitoring and detection systems have been published, but research on their implementation on embedded systems is generally limited. A summary of various embedded fire detection systems is presented in Table I. and a comparison is performed in terms of the number and type of sensors used, the classification algorithms and the implementation platforms. The type of sensors utilized includes environmental sensors, such as temperature, humidity as well as image sensors. The classification algorithms used are based on the fuzzy logic technique (FZ), if-then rules, Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs) and the Dempster-Shafer theory (DS). The algorithms have been implemented on the Arduino Uno, Raspberry Pi, Mica Z, Beaglebone and MSP430 platforms.

A fuzzy logic-based multi-sensor fire detection system was described in [2] and CNNs were trained for indoor and outdoor scenarios. The system used multiple fire signatures including flames, smoke and heat, as well as images from surveillance cameras. It was implemented and tested on a Beaglebone microprocessor, with an accuracy of 94% for the CNNs algorithm and 90% for the FZ algorithm. A system, based on an Arduino Uno, was also designed in [5] based on FZ, to identify the existence of fire. Data was collected from flame, temperature and smoke sensors.

A Wireless Sensor Network (WSN) was implemented in [4] using multiple sensors for indoor fire detection. Smoke, gas, and temperature sensors were used to detect fire on a Raspberry PI. The energy consumption of the deployed sensors was also computed, with a minimum of $0.5mW$ and a maximum of $60mW$ per hour reported.

A low-power sensor node for early detection and monitoring of fire was evaluated in [3]. Two algorithms were developed on an MSP430 microcontroller (MCU) using a temperature and humidity sensor. The first algorithm used a comparison method, while the second was based on DS theory. The same algorithms were utilized in [6] on a low-power ATmega1281 processor. The first was based on a threshold method, using temperature, humidity and light sensors. The second used DS with the nodes being equipped with temperature and humidity sensors. A system to detect forest fires was developed in [1], connecting WSN with ANNs. Low-cost sensor nodes, including temperature, light, and smoke sensors were used and the collected data was encoded as an input to ANNs. The accuracy ranged between 87% to 99% for different scenarios, while the power consumption of the system was not reported.

A CNN was used in [7], [8] for fire detection in surveillance videos. Although this work improved the accuracy of fire detection events, a high rate of false warnings was observed. Moreover, the model size was not suitable for limited-memory embedded systems.

TABLE I
SUMMARY OF THE STATE OF THE ART

Papers	*Sensors*	*Type of sensors*	*Classification*	*Implement.*
[1]	3	Env.	ANN	Mica Z
[2]	4	Env. & Image	FZ & CNN	Beaglebone
[3], [6]	3	Env.	DS theory	MSP430
[5]	3	Env.	FZ	Arduino Uno
[4]	3	Env. & Image	if-then rules	Raspberry Pi
Own work	2	Env.	MLP	Cortex-M4

Previous studies [1], [2], [5], [3] used smoke sensors or a combination of CO_2, temperature and humidity sensors to monitor or detect fire, however these are not sufficient factors to predict actual fire incidents as they could generate false warning signals [5]. Approaches, such as in [4], [2], utilized image and environmental sensors for fire prediction without any constraints in memory requirements and power consumption.

III. System Architecture

This paper introduces a low-power fire monitoring and detection system using an MLP, compact enough that it can be executed on a system with limited memory resources. This section contains details on the proposed MLP model, the hardware and software implementation as well optimizations that can be used to improve the accuracy of the model.

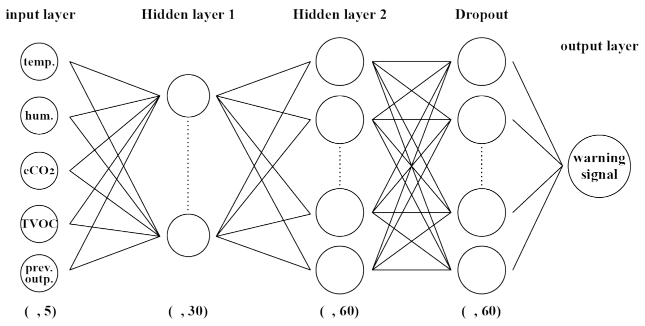

Fig. 1. The proposed MLP model

A. Multilayer Perceptron

ANNs can model complex linear and nonlinear problems and make predictions with a high accuracy [9]. The perceptron [10] is a binary classification algorithm that determines whether inputs belong to a specific event. An MLP is a group of perceptrons that can be represented as layers, able to make predictions for complex problems.

The proposed MLP model consists of five inputs, temperature, humidity, eCO_2, TVOC and the output from the previous execution of the MLP model, two hidden layers and an output with binary values, that describe whether a fire incident has been detected. Different approaches with a variable number of hidden layers and units have been evaluated but as the number of layers and units increased, the accuracy remained at the same levels. Thus, two hidden layers are used. The first contained 30 units, while the second 60 units. In both hidden layers, the rectified linear unit was used as an activation function as it had the best convergence performance. For the output layer, the sigmoid function was used as it was a supervised classification problem with only two values, zero for no fire and one for fire. The overall model is illustrated in Fig. 1.

The MLP was trained using the Adam optimizer for 100 epochs with a learning rate of 5×10^{-4} and a batch size of 32 samples. Subsequently, an L2 penalty of 0.05 was applied on all hidden layers for regularization, as well the dropout regularization with a value of 0.4 before the output of the MLP, to improve its accuracy and to avoid over-fitting.

B. Implementation

The proposed model was implemented on a low-power platform that consisted of an STM32L496 MCU and two environmental sensors, BME680 and CCS811. The MCU is based on a high performance ARM Cortex-M4 32-bit RISC core, operating up to $80MHz$. Cortex-M4 features a Floating-point single-precision unit which supports all ARM single-

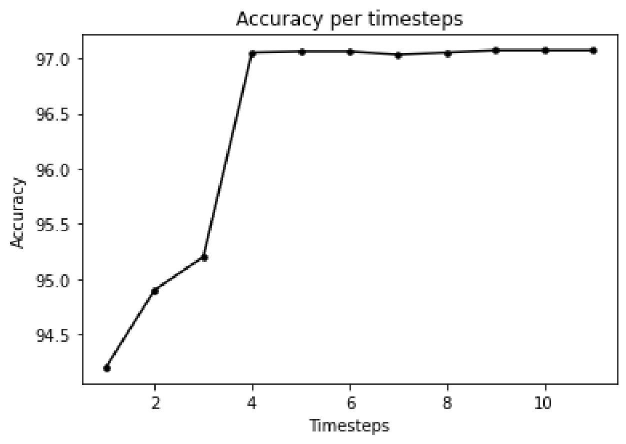

Fig. 2. Accuracy of the model in each timestep

precision data-processing instructions and data types. It also implements a full set of DSP instructions and has $1Mb$ of Flash memory and $320Kb$ of SRAM, split between $256Kb$ of SRAM1 and $64Kb$ of SRAM2. The two sensors are connected on the same I^{2}C bus. The core includes different low-power modes to reduce power consumption when the MCU does not execute, including sleep mode, low-power run and sleep modes, stop 0, 1 and 2 modes, a standby mode and a shutdown mode [11].

BME680 is a gas sensor measuring relative humidity, barometric pressure, ambient temperature and gas (VOC). In order to reduce the consumption of the sensor, the gas and pressure measurements were deactivated. The current consumption of the sensor for temperature and humidity measurement is $2.1\mu A$. CCS811 is a digital ultra-low-power multi-pixel air quality and gas sensor that provides measurements for equivalent calculated carbon-dioxide (eCO$_2$) and Total Volatile Organic Compounds (TVOC). CCS811 supports multiple measurement modes that have been optimized for low-power consumption during an active sensor measurement and idle mode, extending battery life in portable applications. The maximum current consumption of CCS811 is $30mA$ at $3.6V$.

For the experiments of this paper, the two environmental sensors operated on the same $3.3V$ supply as the MCU. The input data from BME680 was stored as float values while from the CCS811 as unsigned integers. Training a Neural Network involves increased data sizes stored in memory and a processor that can support operations with an increased computational complexity. As the MCU used has limited resources, the training phase could not be performed on it. An external node was used instead. Subsequently, the pre-trained MLP model was converted into highly optimized math C code using the Cube AI library, while the complete model was executed directly on the SRAM of the system.

C. Optimizations

The initial implementation included an MCU that operates always in active mode, environmental sensors that draw raw data every second and a neural network that is executed in the same time interval as well, using as an input the measurements of the two environmental sensors. With this approach, the system consumed $61.39mW$ on average, with an accuracy of the MLP model of approximately 94.2%. This section describes different optimizations that can be used to reduce the power consumption and improve the accuracy of the model. A significant percentage of the observed speedup for the proposed algorithm is the result of the combination of these techniques.

1) Chain of MLP models: To improve the prediction accuracy, a chain of MLP models is proposed that use as a feature the output from the previous state and the measurements of environmental sensors from the current state as well. The process is described by the following equation:

$$OutMLP_t = MLP(OutMLP_{t-1}, T_t, H_t, (CO_2)_t, TVOC_t), \quad t \geq 1 \tag{1}$$

where T is the temperature, H the humidity and t the number of the previous states or timesteps. The size of the MLP chain depends on the timesteps and is proportional to t. Fig. 2 describes the relationship of the accuracy to the number of timesteps. Four timesteps have been selected for the proposed system since in that case the highest change of value was achieved in terms of accuracy. As the timesteps increase, the model becomes slightly more accurate but with an increased execution time, as more MLP models are used. Also, using the prediction of the previous state as an input to the next state, could reduce the accuracy of the model, as a prediction error is entered from the previous state. Nonetheless, false prediction from the previous state in combination with data from environmental sensors of the current state do not appear to affect the overall performance of the model. Additionally, and in order to minimize the cost of loop overhead and increase the code efficiency, loop unrolling was applied, in the C code for each timestep of the MLP chain.

2) Quantization: Typically, the values of weights of an ANN are stored as 32-bit floating-points. Quantization reduces the number of bits that are used to store the values of weight and activation functions of ANNs, converting the floating point values to fixed-point integers. The operations between weights are faster as the number of bits is smaller but at the expense of the model's accuracy. Thus, there is a tradeoff between the number of bits that represent weights and the accuracy of the model and some information may be lost [12]. A suitable size of a quantized model with a small memory footprint and a good classification performance is 8 bits [13]. This paper also uses a value of 8 bits for quantization, however, experimentation with different quantization sizes can be also performed. To map float values to integers, two parameters are only required. It is defined as:

$$q_{uint8} = \frac{r_{float32}}{s_{float32}} + z_{uint} \tag{2}$$

where q is the quantized representation, r the real value, s the scale of initial data and z the offset.

3) CPU-specific optimizations: The use of hardware interrupts can significantly reduce power consumption. Initially the MCU is kept in sleep mode. When the value of eCO_2 exceeds a predefined threshold, an interrupt signal is emitted in order to wake up the MCU. The system returns back in sleep mode as soon as the measurement is lower than the threshold. The eCO_2 concentration is used as a threshold since the first warning signal in the case of fire is usually the presence of smoke and an increase in temperature and TVOC is only observed afterwards. Different experiments were performed to determine the appropriate value of the eCO_2 threshold. Fig. 3 presents the values of eCO_2 every 10 seconds for approximately 120 hours of measurements. The dark line describes the scenario where a fire has broken out with a minimum value of eCO_2 concentration of approximately $1500ppm$ on average, while the gray line represents an alternative scenario with a maximum value of eCO_2 concentration of approximately $700ppm$ on average. This paper proposes as a threshold the mean value of the average minimum eCO_2 concentration when a fire has broken out with the average maximum value of eCO_2 concentration when there is no fire incident. Thus, the selected threshold was $1100ppm$.

For the inactive/sleep period of the MCU, the Stop 2 power-mode of operation has been used in order to achieve the lowest-power consumption while retaining the content of SRAM and registers. In this mode, the current consumption is $2.57\mu A$. In run mode, the MCU supports dynamic voltage scaling to optimize power consumption. The voltage from the main regulator that supplies the logic (Vcore) can be adjusted according to the system's operating frequency. The CPU clock of STM32L496 can be executed with a frequency of up to $80MHz$, but for the implementation of this paper a frequency of $4MHz$ was chosen, to reduce the power consumption and minimize latency for accesses to the Flash memory without a significant impact on the execution time of the model.

4) Memory accesses: When reading data from the Flash memory, latency can be introduced in the form of wait states, depending on the frequency of the CPU clock and the internal voltage range of the device. In the case of STM32L496, there are no wait states when the CPU clock is executed at up to $6MHz$ for a supply voltage of $1V$. The SRAM1 and SRAM2 areas can both be addressed by the CPU at a maximum clock frequency without a wait state.

The Flash memory interface of STM32L496 includes a $256B$ data cache memory with 8 cache lines of 4×64 bits each. When data is requested by the CPU, frequently used data lines can be stored in the cache in order to accelerate code execution by enabling a data cache enable (DCEN) bit in the Flash access control register. For the proposed model, the output of MLP for each timestep was stored in the cache memory. The use of the cache when there are no wait states for accessing the Flash memory has no effect on the performance of the algorithm. However, according to [14], the cache should lower the power consumption of up to 20%, since accesses to the cache needs significantly less current compared to accesses to the Flash memory.

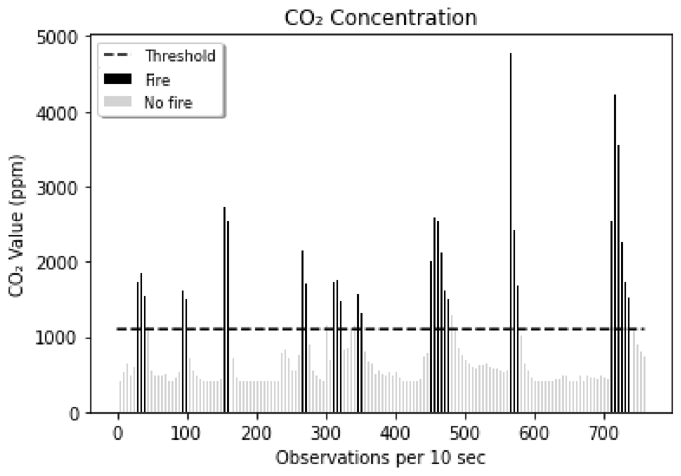

Fig. 3. CO_2 concentration and threshold value

Fig. 4 presents a flow diagram of the proposed algorithm. The process starts by initializing the MCU, restoring the static RAM and initializing the two environmental sensors. After initialization, the gas sensor acquires eCO_2 concentration measurements every $10sec$, the MCU enters sleep mode and the interrupts are enabled. In sleep mode, the main regulator that supplies the core of the MCU is disabled and the Flash memory is powered down while retaining the content of both SRAM and registers. Furthermore, all clocks for the power supply of digital peripherals are stopped and only two low-speed clocks are running. The system remains in sleep mode with the minimum power consumption until the concentration of eCO_2 exceeds a set threshold. When that happens, the interrupt from the CCS811 will enable the MCU from sleep mode, while input data for the temperature, humidity, eCO_2 and TVOC will be available. To detect the abrupt changes in the measurements, the lowest data update rate is chosen for the two sensors. Thus, the sampling rate was changed from $10sec$ to $330msec$ as the lowest data update rate of BME680 is $330msec$. When the system executes the process for fire detection the interrupts are disabled to avoid an unexpected interruption of the program. The execution of the first MLP of the chain uses as input for the variable $OutMLP_{t-1}$ the default value of zero as no fire has been detected, while the next MLP of the chain uses as an input the output of the previous execution. When a prediction is available, the MCU returns in sleep mode, the sampling rate changes to $10sec$ while the interrupts are enabled.

IV. Evaluation

This section presents the evaluation methodology, the evaluation metrics, the power consumption as well as the memory requirements of the proposed system. A comparison of the accuracy of the proposed model to existing approaches is also provided.

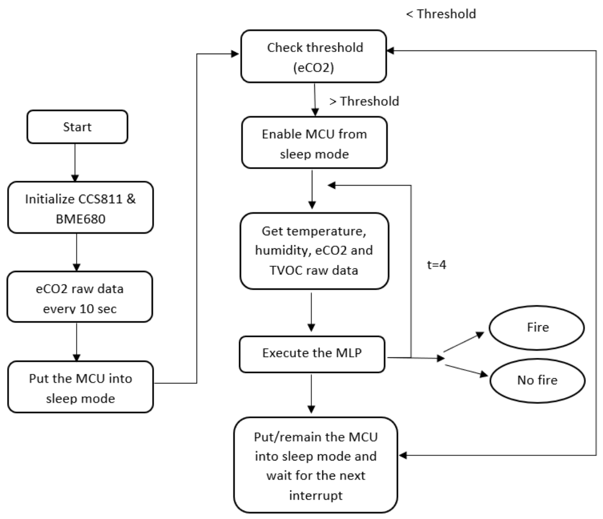
< Threshold
Check threshold (eCO2)
> Threshold
Start
Enable MCU from sleep mode
Initialize CCS811 & BME680
Get temperature, humidity, eCO2 and TVOC raw data
t=4
eCO2 raw data every 10 sec
Fire
Execute the MLP
No fire
Put the MCU into sleep mode
Put/remain the MCU into sleep mode and wait for the next interrupt

Fig. 4. Flow diagram to handle interrupts and detect fire

A. Methodology

A set of data has been collected using the two sensors for room environment monitoring. The raw data, that describe the fire and no fire events, was acquired at regular 10-second intervals. The experiments took place in an office with an approximate temperature of 23^oC and relative humidity of approximately 32%. A minor fire incident was simulated.

In total, 4500 data points have been collected from the temperature, humidity, CO_2 and TVOC sensors, with a precision of two decimals. Additionally, the dataset was balanced as it contained equal instances from the fire and no fire classes.

As the proposed ANN used a gradient descent method as an optimization technique, the input data had to be scaled. The paper applied a standardization method, shifting the distribution of each attribute to have a mean of 0 and a standard deviation of 1.

B. Evaluation Metrics

For the evaluation phase, the k-fold cross-validation technique was used. K-fold is a resampling procedure which divides the initial dataset into k sub-samples of equal size at random. A single sub-sample from the k possible is kept as validation data for testing the model, while the remaining $k-1$ are used for the training phase. A common choice of k is between 5 and 10. For this paper, $k = 10$ was used.

To evaluate the performance of the proposed model, an Area Under The Curve (AUC) - Receiver Operating Characteristics (ROC) curve was used. The ROC is represented with a probability curve that plots the false positive rate on the X axis and the true positive rate on the Y axis. The AUC measures the ability of a classifier to separate properly the dataset between two classes. Also, it is used as a summary for the ROC curve. The acceptable values are in the range $[0-1]$ with the value of 1 indicating that the model is a perfect classifier. The accuracy and $f1$ score, a metric that combines the precision and recall of a classifier into a single metric, are used to compare the proposed model with existing works. Moreover, the sensitivity, a metric that measures the ability of the model to predict true positives of each class, as well as the specificity, a metric that measures the ability of the model to predict true negatives of each class were used for evaluation purposes.

TABLE II
COMPARISON OF MODELS IMPLEMENTATION

Papers	***Accuracy***	***Enviromental Sensors***
[4]	85%	temperature, smoke, gas sensor
[2]	90%	temperature, flame, smoke sensor
[5]	95.3%	temperature, humidity, flame sensor
Own work	97.05%	temperature, humidity, TVOC, CO_2 sensor

C. Experimental Results

Fig. 5 presents the AUC - ROC curve of two variations of the proposed method using 4 timesteps. The solid line describes the model variation represented with 32 bits, while the dashed line the model with 8 bits representation. The performance of both variations using the AUC metric was very close, with a small decrease for the model with 8 bits, with values of 0.99 and 0.97 respectively. As expected, the variation with the fewest bits had a lower ability to separate the two classes, but with a smaller size of the weights' file.

Table II shows a comparison between the proposed model and the literature. Different data points were used in the presented implementations from the literature but the comparison made in terms of the effectiveness of the models as well as on the sensors used. The model introduced in this paper, with an 8-bit representation of weights and a k-fold validation, had an accuracy of 97.05% with an $f1$ score of 97%, a sensitivity of 97.1% and a specificity of 96.5%, among the highest between the three presented implementations. The $f1$ score was affected by an incorrect classification of class 0 values to class 1 (fire occurrences). In these cases, the system was creating false alarms, but this type of error does not have a significant consequence as in the opposite scenario, the non-activation of alarm in case of fire. Subsequently, Fig. 6 illustrates the training and validation average accuracy of the model, using the k-fold cross-validation. As can be seen after epochs 80, a remarkable accuracy has been achieved on both the training and validation dataset. Moreover, as can be seen from Table II, the introduction of the eCO_2 and TVOC values improved the accuracy of the model compared to the other implementations.

D. Energy and Memory Requirements

The total power consumption can be defined as the sum of the power expenditure during the active mode, divided into data acquisition and processing, as well as the sleep mode. In active mode, the MCU and the two environmental sensors are switched on, having a power consumption of $60.8mW$ on average with an energy cost of $80.5mJ$. When the system remains in sleep mode, the MCU operates on $13.7\mu A$ while the CCS811 sensor consumes only $4.6mA$ until the value of

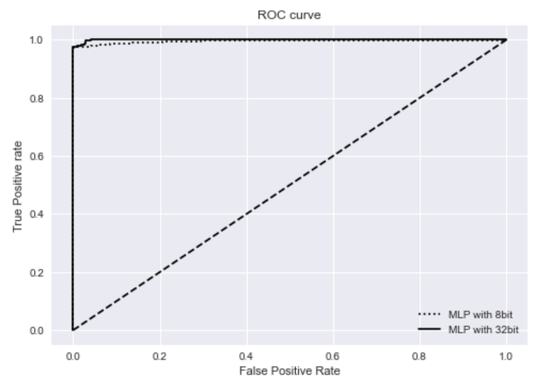

Fig. 5. AUC - ROC Curve of proposed model

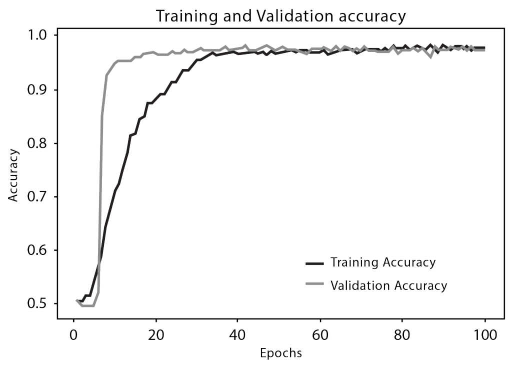

Fig. 6. Accuracy at each epoch for train and validation dataset

the eCO_2 concentration crosses the set threshold or the 10-second interval has elapsed. In total, the sleep mode power consumption is $15.2mW$ with a peak value of $59.4mW$ every 10 seconds. Information on the power consumption for implementations from the literature reviews was generally not available, thus a comparison with these works was not feasible.

The execution of the MLP has a computational complexity of $1.8K$ multiply-accumulate operations for each input and the execution time for each decision is approximately $43msec$. When the system is in active mode, it consumes on average $2.31mW$ with the MLP algorithm requiring approximately $0.1mJ$ for each prediction. Thus, using a $5100mAh$ off-the-shelf battery, with the system in sleep mode with interrupts and an eCO_2 measurement every 10 seconds, the system can operate for over 37 days without any human intervention.

The memory requirements of the proposed system depend on three factors, the input data from the sensors, the weights of the MLP model and the intermediate results between the hidden layers. The input data from the temperature and humidity sensors is float, while from eCO_2 and TVOC is $uint16\_t$. Consequently, the total memory requirements are 16 bytes, while the weights of the MLP model require $25.65kB$ of Flash memory.

V. Conclusions

This paper proposed a low-power embedded system for fire monitoring and detection using an interrupt-based algorithm with a machine learning model. The MLP algorithm was used with two hidden layers and a quantized model represented with 8 bits. Compared to the other implementations in the literature, the proposed approach had a remarkable accuracy close to 97%, an ultra-low-power consumption and a model size that requires only 4.9% of the $1Mb$ Flash memory.

In terms of future work, it will be interesting to integrate an image sensor on a low-power embedded processor and using a machine learning model make predictions for fire incidents while keeping a low-power consumption. In addition, the design and implementation of a new CO_2 sensor with lower power consumption would be helping to achieve a better result.

Acknowledgment

This work is co-funded by the European Union (EU) within the AMANDA project under grant agreement number 825464. The AMANDA project is part of the EU Framework Programme for Research and Innovation Horizon 2020.

References

[1] H. Soliman, K. Sudan, A. Mishra, "A Smart Forest Fire Early Detection Sensory System, Another Approach of Utilizing Wireless Sensor and Neural Networks.", In Proceedings of the IEEE Sensors 2010 Conference, Kona, HI, USA, 1–4 November 2010.

[2] R.A. Sowah, K. Apeadu, F. Gatsi, K.O. Ampadu, B.S. Mensah, "Hardware Module Design and SoftwareImplementation of Multisensor Fire Detection and Notification System Using Fuzzy Logic and ConvolutionalNeural Networks (CNNs)." Journal of Engineering ,vol. 2020, February 2020.

[3] S.R. Vijayalakshmi, S. Muruganand, "Real Time Monitoring of Wireless Fire Detection Node", Procedia Technology 2016, vol. 24, p. 1113-1119.

[4] F. Saeed, A. Paul, A. Rehman, W.H. Hong and H. Seo" IoT-Based Intelligent Modeling of Smart Home Environment for Fire Prevention and Safety", Journal of Sensor and Actuator Networks 2018, 7, 11.

[5] B. Sarwar, I.S. Bajwa, S. Ramzan, B. Ramzan, M. Kausar, "Design and application of fuzzy logic based firemonitoring and warning systems for smart buildings." Symmetry 2018, 10, 615, November 2018.

[6] A. Díaz-Ramírez, L.A. Tafoya, J.A. Atempa, P. Mejía-Alvarez, "Wireless Sensor Networks and Fusion Information Methods for Forest Fire Detection", Procedia Technology 2012, vol. 3, p. 69-79, May 2012.

[7] S. Frizzi, R. Kaabi, M. Bouchouicha, J. Ginoux, E. Moreau and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 877-882, 2016

[8] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," in IEEE Access, vol. 6, pp. 18174-18183, 2018.

[9] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016

[10] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization", Psychological Review, Vol. 65, No. 6, 1958.

[11] STMicroelectronics: RM0351 reference manual (2021). https://www.st.com/resource/en/reference_manual/dm00083560-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[12] W. Roth, G. Schindler, M. Zohrer, L. Pfeifenberger, R. Peharz, S. Tschiatschek, H. Froning, F. Pernkopf, Z. Ghahramani, "Resource-Efficient Neural Networks for Embedded Systems", 2020.

[13] V. Vanhoucke, A. Senior, M.Z. Mao, "Improving the speed of neural networks on CPUs", In Proceedingsof the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, December 2011.

[14] STMicroelectronics: STM32 power mode examples - Application note (2019). https://www.st.com/resource/en/application_note/dm00237631-stm32-power-mode-examples-stmicroelectronics.pdf