Ingeniería y Universidad
ENGINEERING FOR DEVELOPMENT

Vol. 29
Continuous Publication
2025

ISSN: 2011-2769 (Online) | ISSN: 0123-2126 (Print)

Artículos

Performance Analysis of a Convolutional Neural Network for Pneumonia Detection on an Embedded AI System*

Análisis del rendimiento de una red neuronal convolucional para la detección de neumonía en un sistema de IA integrado

DOI: https://doi.org/10.11144/Javeriana.iued29.pacn

Yasser Abd Djawad [a]
Universitas Negeri Makassar, Indonesia
yasser.djawad@unm.ac.id
ORCID: https://orcid.org/0000-0003-0739-5968

Dary Mochamad Rifqie
Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/0000-0002-5376-574X

Ridwansyah
Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/000-0003-3540-7364

Supriadi
Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/0000-0002-4090-1386

Saharuddin Ronge Sokku
Electronics Department, Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/0009-0005-9310-7951

Andi Baso Kaswar
Electronics Department, Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/0000-0002-8581-7807

Ma'ruf Idris
Electronics Department, Universitas Negeri Makassar, Indonesia
ORCID: https://orcid.org/0000-0001-7061-1569

Received: 20 december 2023
Accepted: 19 june 2025
Published: 05 august 2025

Abstract:

Objective: The increasing use of artificial intelligence (AI) in various fields has increased the need for a large amount of data. A device with adequate computational power is required to manage the data and produce an output with high processing speed and satisfactory accuracy. Moreover, the use of several embedded-system devices for neural networks (NNs) is constrained by low processor and memory capacity. Several embedded-system devices with improved processor capabilities have been developed for NN data processing. *Materials and method*: In this study, the capabilities of an embedded-system device for NNs in health applications was analyzed; namely, the detection of X-ray images of patients with pneumonia using a convolutional neural network (CNN) was tested. Two-dimensional CNN architectures with various parameters, including color depth, layers, fi lters, kernels, and quantization, were employed. The outcome was expressed in terms of accuracy, inference time, RAM, and flash consumption. *Results and discussion:* The results revealed a significant positive association between all output metrics and the number of filters. However, in some situations, the RAM and flash utilization of the embedded system exceeded its capacity, making it unusable. Thisfinding indicates that the inference time and memory are influenced by the number of layers, filters and quantization. *Conclusion*:Thus, the use of embedded-system devices for CNN can be done with proper hyperparameter tuning.

Author notes

[a]Corresponding author. E-mail: yasser.djawad@unm.ac.id

Keywords: Pneumonia Classification, Embedded AI System, Inference Time, Quantization.

Resumen:

Objetivo: el creciente uso de la inteligencia artificial (IA) en diversos campos ha aumentado la necesidad de una gran cantidad de datos. Se necesita un dispositivo con la potencia de cálculo adecuada para gestionar los datos y producir un resultado con una alta velocidad de procesamiento y con una precisión satisfactoria. Además, el uso de varios dispositivos de sistemas integrados para redes neuronales (NN) se ve limitado por la escasa capacidad del procesador y de la memoria. Se han desarrollado varios dispositivos de sistemas embebidos con capacidades de procesador mejoradas para el procesamiento de datos de redes neuronales. *Materiales y método*: en este estudio se analizaron las capacidades de un dispositivo de sistema integrado para redes neuronales en aplicaciones sanitarias; en concreto, se probó la detección de imágenes de rayos X de pacientes con neumonía, mediante una red neuronal convolucional (CNN). Se emplearon arquitecturas CNN bidimensionales con diversos parámetros, como profundidad de color, capas, filtros, núcleos y cuantización. Los resultados se expresaron en términos de precisión, tiempo de inferencia y consumo de RAM y flash. *Resultados y discusión:* los resultados revelaron una asociación positiva significativa entre todas las métricas de salida y el número de filtros. Sin embargo, en algunas situaciones, la utilización de RAM y flash del sistema embebido superó su capacidad, haciéndolo inutilizable. Este hallazgo indica que el tiempo de inferencia y la memoria se ven influidos por el número de capas, filtros y cuantización. *Conclusiones*: así pues, el uso de dispositivos con sistemas embebidos para CNN puede realizarse con un ajuste adecuado de los hiperparámetros.

Palabras clave: clasificación de la neumonía, sistema de IA integrado, tiempo de inferencia, cuantización.

Introduction

Pneumonia is a lung infection caused by bacteria, viruses, mycobacteria or fungi [1]. Common symptoms found in pneumonia patients are coughing, fever, shortness of breath, nausea, and vomiting [2]. Pneumonia diagnosis is made by listening to the sound of breathing, which is usually heavy and rumbling. The next step is to use X-ray technology to capture photographs of the chest region to observe the lung area. The location of the infection in the lungs is determined from the collected images to determine the area and degree of dissemination. Additionally, a patient's blood is typically examined in a laboratory to establish the cause.

One of the main causes of pneumonia is pollution [3], and in recent years, pneumonia has been caused by the COronaVIrus Disease of 2019 (COVID-19)virus [4]. Pneumonia is a malignant disease that can lead to death. Pneumonia patients range in age from children to elderly individuals. In developing countries, patients with pneumonia do not receive effective treatment. The lack of effective treatment potentially contributes to the high mortality rate associated with this disease [5]. However, in developing countries, diagnostic equipment is expensive. Devices with high computational capabilities are available only in large hospitals in large cities. Therefore, a tool to assist doctors in detecting pneumonia that is inexpensive, does not require complex treatment, can be used easily, and allows the device to be used as a data source, which is the core concept of edge computing (EC), is needed.

Pneumonia has been identified in multiple studies employing patient X-ray images and various methodologies. Several image-processing techniques have been applied, such as integrated filters, enhanced contrast, edge detection [6], and texture analysis [7]. Another approach is to use the Earth Mover's Distance (EMD) to identify the infected lungs and normal lungs using the distance between two probability distributions [8]. Furthermore, to assist this analysis with a computer-aided diagnosis system, several studies have been conducted, including machine learning (ML) [9], deep learning (DL) [10]–[13] and ensemble DL [8], [12], [14], to improve the performance of CNNs. This ensemble DL combines several well-known CNN model outputs during the experimental phase. Another CNN technique based on semantic and instance segmentation, called the mask region-based convolutional neural network (Mask RCNN), was applied to pneumonia X-ray images [15]. However, all these CNN approaches require prediction, detection, and classification. On the hardware side, this condition calls for additional components, such as the graphical processing unit (GPU) processor, to accelerate and alter images in memory [16], [17]. In this study, computing was performed on computer, and the deployment of DL to an embedded-system device that does not have GPUs.

Several previous studies have used microcontrollers for machine learning (ML). A support vector machine (SVM), k-nearest neighbor (kNN), and decision tree were deployed to the STM32 Nucleo Board [18]. To provide an automatic intelligent system, kNN was applied to the TI CC1110F32 microcontroller [19]. However, running DL on embedded systems that have limited capacity is still complicated because DL applications require quick and precise data processing owing to the vast volume of input data [20]. The physical size [21], data processing speed [22], memory usage [23], and electric power consumption [24] are important factors in the development of embedded systems for DL. Several efforts have been made to make complex calculations more efficient and reduce memory usage in DL by making adjustments to the DL architecture [25]–[28] or changing variable parameters from 32-bit floats to lower parameter types, also known as quantization [29], [29], [30].

Currently, sizable data are commonly sent from edge devices and processed in the cloud. The results are then sent back to the edge. However, this process is time-consuming. Therefore, a device that performs processing directly on the edge is currently under development, which would save time [31]. This study offers a further option for CNN applications in embedded systems by performing a variety of analyses of pneumonia X-ray images on a number of input and output characteristics to determine whether they can be used in embedded systems designed specifically for AI. This analysis was also intended to provide a general overview of the issues that may arise when a CNN is implemented in an embedded system. The selected application was a medical-related application because current medical devices are required to be portable devices that can be used anywhere.

Methodology

Dataset

The collected dataset was uploaded to the Edge Impulse Studio. The Edge Impulse Studio is an integrated development environment for creating models of neural networks until deployment to target hardware, which already integrates Python programming languages and TensorFlow. X-ray images were taken from datasets that were shared in previous research [32]. The dataset contained X-ray images in JPEG format, and the images were divided into two categories: normal and pneumonia. The dataset that was randomly selected for this study consisted of 360 images for the normal category and 360 images for the pneumonia category (bacterial and viral types). Of the 720 images, 576 (80%) were used for the training dataset, and 144 (20%) were used for the test dataset. Preprocessing started with scaling, where the images were sized to 96 × 96 pixels (fit shortest). Then, the images were normalized to a float value using the following formula:

$$normalised\ pixel = \frac{pixel\ value}{255} \tag{1}$$

In the Edge Impulse Studio, to reduce the color depth, ITU-R BT.601 conversion (Y only) was performed to convert RGB to grayscale; for red green blue (RGB) color, the RGB565 format was used. The color depth was reduced, which produced 9,216 grayscale features and 27,648 RGB features, as shown in Figure 1. The Uniform Manifold Approximation and Projection for dimension reduction (UMAP) algorithm (UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — Umap 0.5 Documentation, n.d.) was used to visually analyze the appearance of the dataset to determine its distribution, as illustrated in Figure 2.

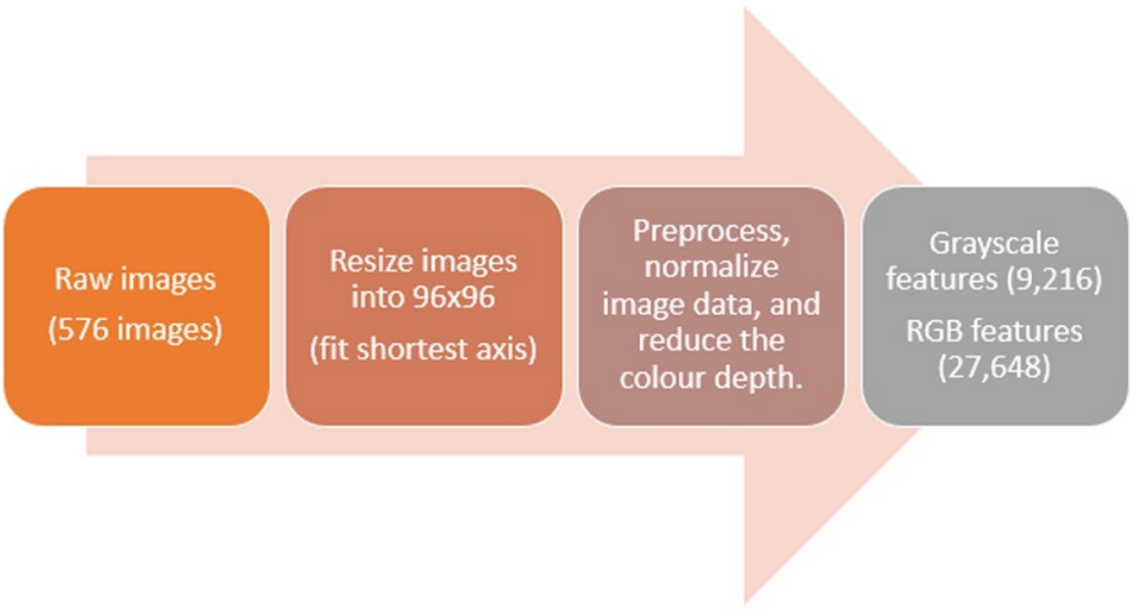

Raw images
(576 images)
Resize images into 96x96
(fit shortest axis)
Preprocess, normalize image data, and reduce the colour depth.
Grayscale features (9,216)
RGB features (27,648)

FIGURE 1
Preprocessing for training dataset images
Source: Own elaboration.

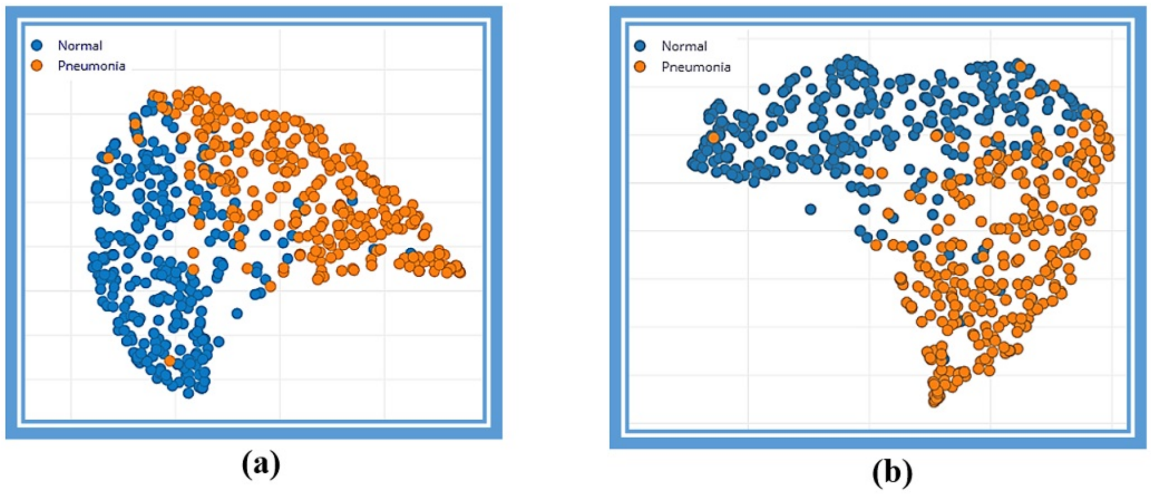


FIGURE 2
Visualization of the image dataset using the UMAP algorithm: (a) grayscale and (b) RGB
Source: Own elaboration.

System

The hardware used was an Arduino Nano BLE 33, as shown in Figure 3. The Arduino Nano BLE 33 contains a 64 MHz nRF52840 32-bit CPU, 1 MB of flash memory, and 256 KB of RAM. Arduino Nano BLE 33 was chosen because it can perform NN processing using TensorFlow and TinyML with a minimal hardware environment.

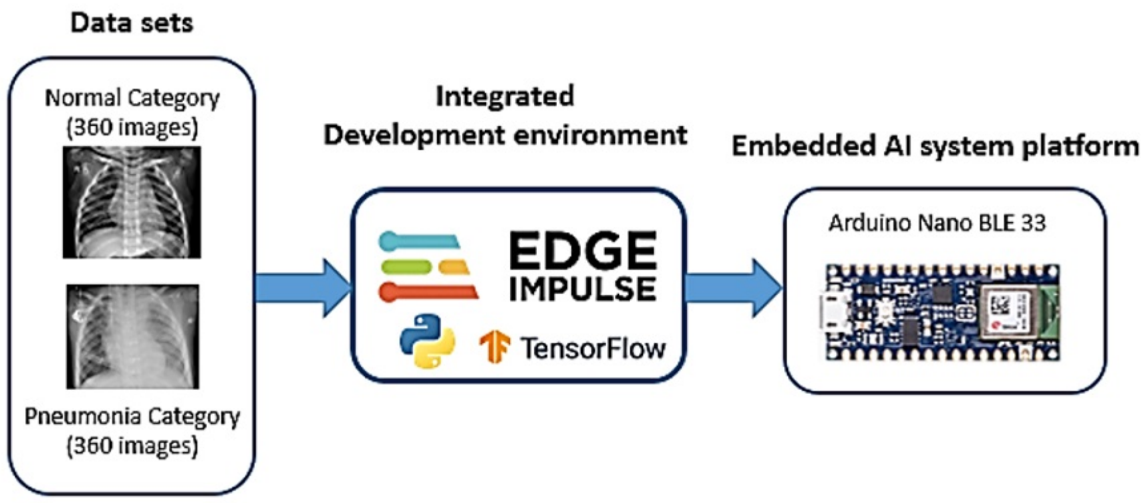

Data sets
Normal Category (360 images)
Pneumonia Category (360 images)
Integrated Development environment
EDGE IMPULSE
TensorFlow
Embedded AI system platform
Arduino Nano BLE 33

FIGURE 3

General diagram of the performance analysis of CNNs in the embedded AI system platform

Source: Own elaboration.

Convolutional Neural Network (CNN)

As depicted in Figure 4, the CNN consists of three constituent layers: the convolutional layer, pooling layer and fully connected (FC) layer. In the convolutional layer, the convolution process is performed on the basis of the previous layer using the kernel to display a feature map [33], which can be described as follows:

$$x_j^i = f(\Sigma_{i\in M_j} x_i^{I-1} * k_{ij}^I + b_j^I) \tag{2}$$

where *x* is the output of the current layer, x^{I-1} is the previous layer, *k* is the kernel and *b* is the bias.

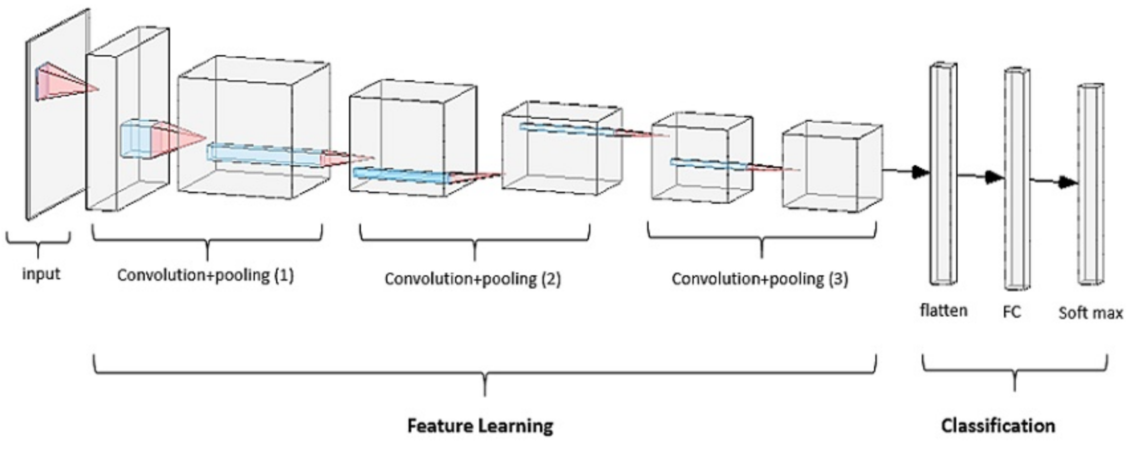


FIGURE 4

Architecture diagram of the 2D CNN, which consists of 3 convolution and pooling layers and 1 flattening layer

Source: Own elaboration.

The pooling layer reduces the resolution of the input in the previous layer, thereby reducing the spatial size of the representation. Reducing the resolution is intended to speed up the computation. The required parameters are the filter size, stride, and max or average pooling.

Quantization

Quantization was applied to the CNN model to facilitate deployment on low-resource embedded systems. Quantization means reducing the precision of a model's weights and activation. In this study, 32-bit floating-point numbers were quantized for the weights and activations into 8-bit integers by posttraining quantization (PTQ). Quantization reduces the model size and memory access and improves the computational efficiency of low-precision enabled embedded hardware. The quantization process is illustrated by the equation below:

$$q = int\left(\frac{r}{S}\right) - Z \tag{3}$$

where q is the quantized value, r is the input in real-valued form, S is a scaling factor, and Z is an integer zero point.

A linear symmetric PTQ was employed to quantize the model parameters and activations. This method assumes that the value range is zero-centered; therefore, the need for a zero point is eliminated. Consequently, the implementation complexity is low, and the computational complexity is reduced. Quantization begins with the calculation of a scale factor, which is used to calculate the quantized values. Scale S was calculated via the following expression:

$$S = \frac{max-min}{2^b} - 1 \tag{4}$$

where max and min are the maximum and minimum values of the parameter (such as weights or biases), respectively, and b is the bit width used for quantization.

Metrics

In this study, the parameters used included accuracy, quantization, inference time, RAM utilization, and flash usage. The proportion of accurate estimates throughout the entire dataset is the DL accuracy, and the formula is as follows:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{5}$$

where TP=True Positive, TN=True Negative, FP=False Positive and FN=False Negative. The number of positive data correctly identified by the model was denoted by TP, the number of negative data correctly identified by the model was denoted by TN, the number of positive data incorrectly identified by the model was denoted by FP, and the number of negative data incorrectly identified by the model was denoted by FN. The accuracy reflects the overall performance of the model. Another metric used was quantization. In this study, two types of numbers were evaluated: int 8 and float 32. The inference time is the amount of time it takes for a model to study test data and generate a prediction, and it typically excludes the training time. However, the amount of RAM and flash memory required to accomplish data processing and matching depends on their usage.

Experimental setup

Several experiments were conducted using a 2D CNN by varying the hyperparameters of the CNN. Various combinations of parameters, such as color depth (Grayscale and RGB), number of layers (2 and 3), number of filters (4, 8, 16, 32, 64 and 128), size of kernels (1, 2, 3 and 4) and number types (int 8 and float 32), were applied. Grayscale and RGB images were chosen to determine the effect of color on the inference time. Moreover, the selection of the number of filters, which is a power of 2, and the kernel size are used to determine the effects of the number of filters and kernel size complexity on accuracy, inference time, and memory usage, considering that the memory on the board is limited. Because the models are deployed in resource-constrained hardware and require real-time performance, the use of 8-bit quantized and 32-bit unoptimized models needs to be considered. Furthermore, the results of these adjustments to the parameters, including accuracy, inference time, and the utilization of RAM and flash memory, were evaluated.

TABLE 1
Training settings

Parameters	Description	Value
Number of training cycles	the number of repetitions of the training cycle	10
Learning rate	a tuning parameter in an optimization method that controls the weights of our neural network in proportion to the gradient of the loss and that determines the step size for each iteration.	0.0005
Validation set size	dataset used to train NN to find the best model	20
Dropout rate	during training, a certain set of neurons chosen at random is disregarded.	0.25

Source: Own elaboration.

Results

Numerous output measures were obtained from the training outcomes. The output can also be displayed as a number using a 2-dimensional visual graph, as illustrated in Figure 5. The figure shows the data that were properly guessed as well as the wrongly guessed data. This visualization makes the analysis easier by providing a broad perspective of the position of the data, if shown on a graph.

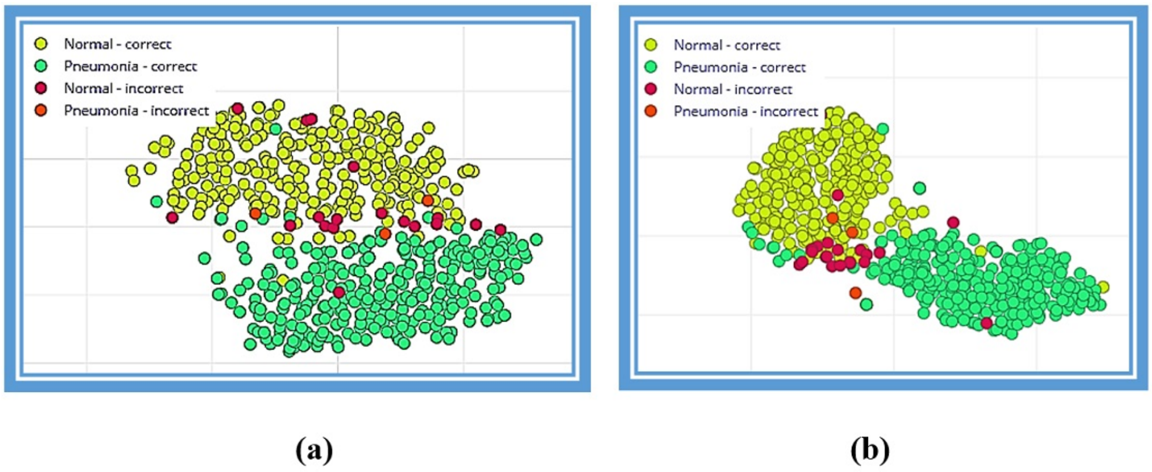


FIGURE 5

Examples of visualization of the full training dataset: (a) 16/32, 2 layers, grayscale, 3 layers, grayscale, kernel size of 3; (b) 16/32, 2 layers, RGB, kernel size of 3

Source: Own elaboration.

The CNN architecture, which consists of two layers and three kernels with grayscale color depth, and tests with various filter pairings are presented in Figure 6. The findings demonstrate that increasing the number of filters in each layer improved the accuracy from 86.1% to 94.8%. Moreover, the inference time exhibited exponential growth. This demonstrates that as the number of filters increases, the computational complexity of the CNN architecture likewise increases, thereby lengthening the inference time. The RAM and flash usage also increased as the number of filters increased for int 8, as shown in Figure 6a. The inference time significantly increased when the number was changed to 32, as shown in Figure 6b. Additionally, it results in a large increase in RAM and flash memory. This reveals that filter 32/64 for both types of numbers and 16/32 for float 32 use more RAM than does 256 K, which means that they cannot be deployed to NANO BLE 33.

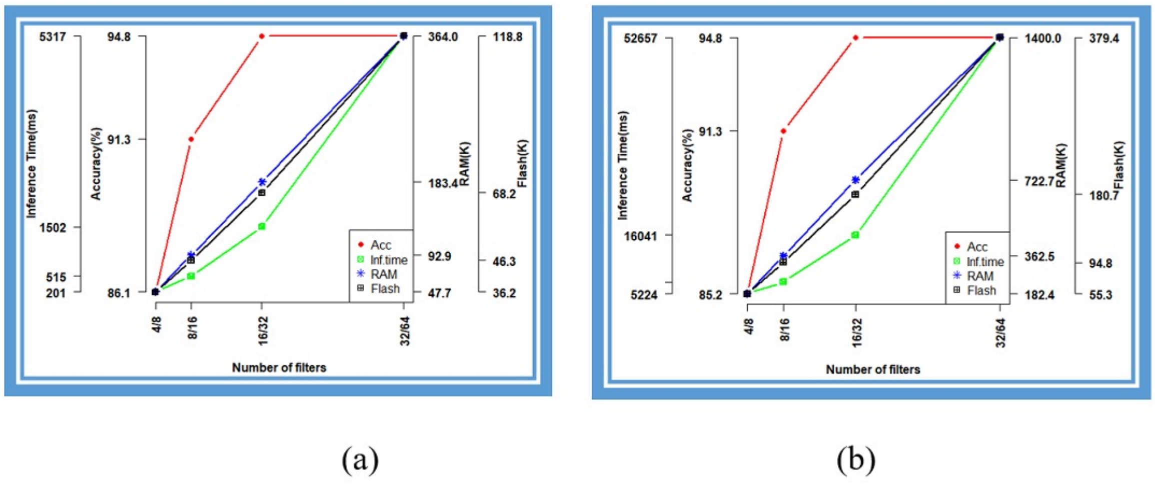
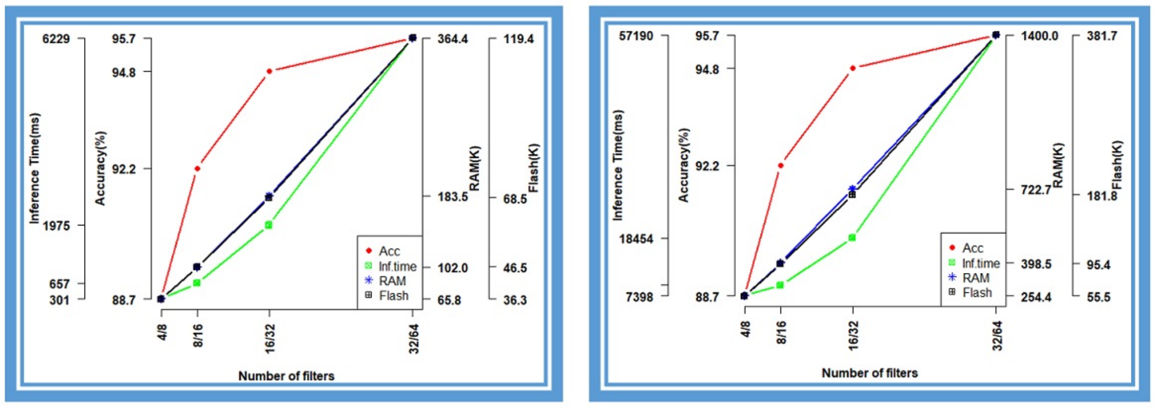


FIGURE 6

2D CNN, grayscale, 2 layers, kernel size of 3 (a) int 8 and (b) float 32

Source: Own elaboration.

The RGB color depth, which has three times as many features as grayscale features do, yields nearly the same results, as shown in Figure 7. The accuracy, inference time, RAM, and flash utilization increased as the number of filters increased. The inference time, RAM, and flash utilization were clearly greater than the grayscale color depth. This is understandable because of the large number of inputs. However, there is no change in the accuracy between the number types int 8 and float 32. The RAM utilization in the 4/8 filter for float 32 is quite close to the NANO BLE 33 RAM's maximum capacity; therefore, it is not advised to use it in this situation according to the RGB input. It is clear from the additional filters in this section that more RAM is needed than the amount of RAM on the NANO BLE 33.



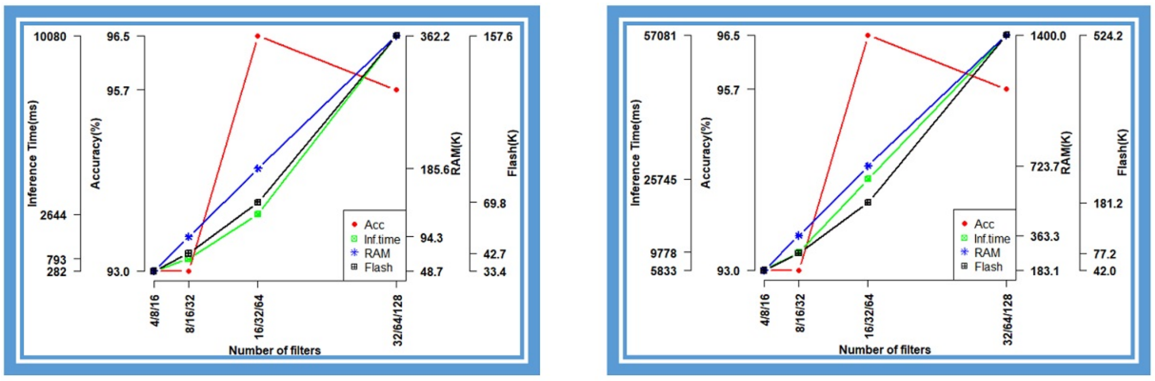
(a) (b)

FIGURE 7

2D CNN, RGB, 2 layers, kernel size of 3 (a) int 8 (it demonstrates that because filter 32/64 uses more RAM than 256K does, and it cannot be deployed to NANO BLE 33) (b) float 32. (It shows that in all cases, this model cannot be deployed since RAM usage is more than 256K)

Source: Own elaboration.

The addition of one layer to the architecture with color depth grayscale also increases the accuracy and other output parameters, as shown in Figure 8. There is an insignificant increase and even a decrease in accuracy when layers are supplemented with a 128 filter. The other output parameters were also increased. Compared with the grayscale architecture using two layers, the inference time is longer. However, the use of RAM and flash memory is similar. There is a significant difference in the inference time, RAM and flash usage when quantization is used without optimization. Similar to adding a layer to the grayscale input, this is seen in various situations but cannot be used with NANO BLE 33 because it uses more RAM than is provided. For example, on float number 32 for type int 8, filter 32/64/128 and all other filters aside from filter 4/8/16.



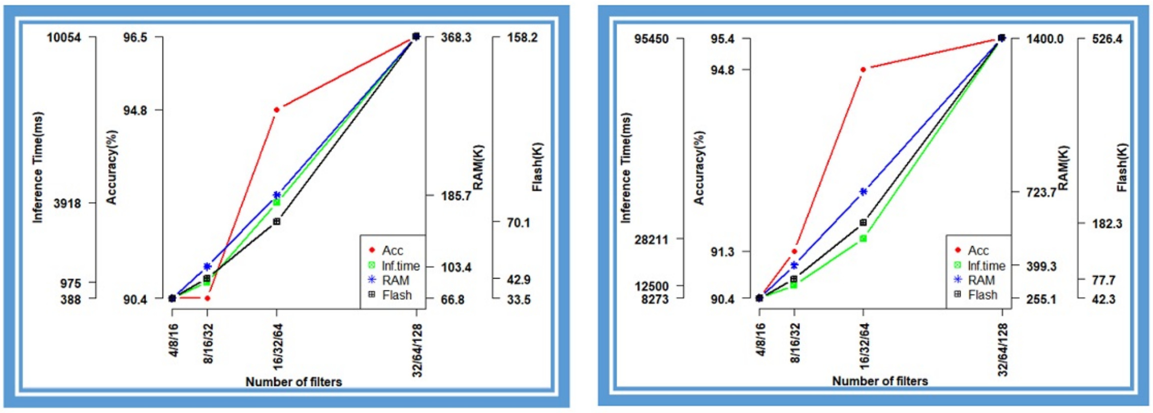
(a) **(b)**

FIGURE 8

The 2D CNN, grayscale, 3 layers, and kernel size of 3 demonstrates that because the 32/64 filter uses more RAM than the 256K filter does, it cannot be deployed to NANO BLE 33 (a) int 8 (b) float 32

Source: Own elaboration.

Figure 9 shows the output using the RGB input and three layers. The same tendency, that is, an increase in all parameters, was also observed when the RGB input was used. In contrast to the grayscale input for quantization, this had a smaller impact. With the same number of screens and grayscale inputs, it is clear from the unoptimized number that the inference time varies significantly. However, compared with the grayscale input with the same number of layers, the RAM and flash utilization did not noticeably change. The RGB input and the three layers exhibit nearly identical patterns for RAM utilization that exceeds the RAM capacity and cannot be used with NANO BLE 33.



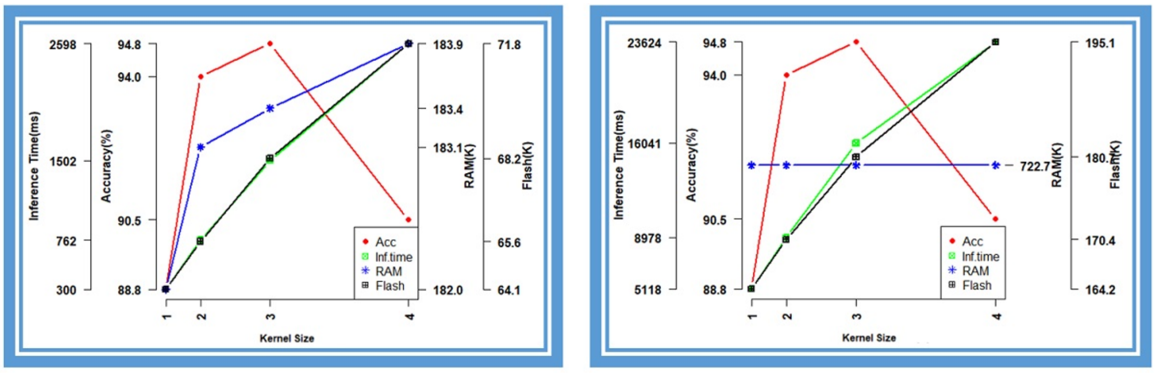
(a) **(b)**

FIGURE 9

The 2D CNN, RGB, 3 layers, and kernel size of 3 demonstrates that because the 32/64 filter uses more RAM than the 256K filter does, it cannot be deployed to NANO BLE 33 (a) int 8 (b) float 32

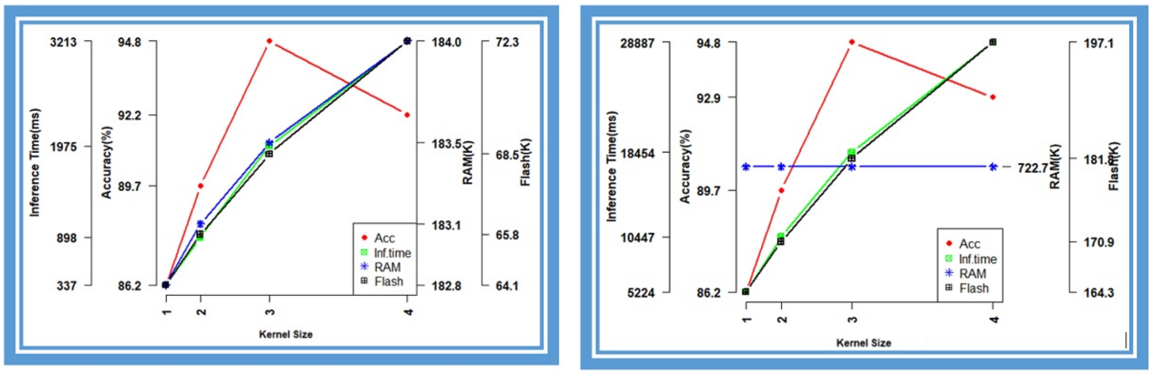
Source: Own elaboration.

The accuracy and use of RAM and flash memory improved little or not at all after the kernel size was increased. There was an even decrease in the accuracy when the kernel size was 4. However, the inference time appears to have significantly increased for both the quantized and unoptimized values, as shown in Figures 10 and 11. The use of a kernel size of 3 resulted in the best performance. Additionally, the use of RAM has no effect on the type of number without optimization (float 32).



(a) (b)

FIGURE 10
Various kernel sizes of 2D 16/32 filter CNN, grayscale, 2 layers (a) int 8 (b) float 32
Source: Own elaboration.+



(a) (b)

FIGURE 11
Various kernel sizes of 2D of 16/32 filter CNN, RGB, 2 layers (a) int 8 (b) float 32
Source: Own elaboration.

Discussion

Reducing inference time and memory usage is essential for embedded AI system applications, such as agricultural and health surveillance edge devices placed in resource-constrained areas, to improve energy usage and increase battery life. Therefore, by adjusting the hyperparameters on the basis of the needs of a given task, resource usage can be optimized without reducing the performance of the device.

To determine the significance of the correlation between the output parameters, a correlation test was performed using Pearson correlation, and the results are shown in Table 2. The table shows that the correlations between accuracy and inference time and between RAM and flash usage are positive in the range of 0.5–0.9, which means that an increase in the number of filters on the layer increases accuracy, indicating a very strong positive correlation with other parameters.

The lowest correlation is seen in the use of accuracy vs. the use of flash on grayscale, type number int 8, and layer 3 (e.g., r=0.558671), and the highest correlation is seen on grayscale, type number float 32, and three layers (e.g., r=0.928159). In layer 2, the correlation results of the 8-bit quantized numbers and 32-bit unoptimized numbers are not significantly different. In layer 3, there is a significant difference between 32-bit unoptimized and 8-bit quantized numbers. However, the use of the 32-bit unoptimized model also results in fairly large memory usage on embedded systems; therefore, it is necessary to consider its use on embedded systems that have a small memory capacity. In this case, the use of an 8-bit model suggests high performance at low computational and energy costs.

TABLE 2
Correlation of increasing filter number with various parameters using Pearson correlation

Layer	Number type	Grayscale			RGB		
		Accuracy vs. Inf. Time	Accuracy vs. RAM usage	Accuracy vs. Flash usage	Accuracy vs. Inf. Time	Accuracy vs. RAM usage	Accuracy vs. ROM usage
2	int 8	0.668808	0.786772	0.760043	0.777116	0.843238	0.840563
	float 32	0.635172	0.782396	0.746015	0.741522	0.850573	0.840341
3	int 8	0.609285	0.736235	0.558671	0.720769	0.741072	0.642097
	float 32	0.928159	0.927557	0.893181	0.795026	0.889575	0.829051

Source: Own elaboration.

Previous research has also examined the application of medical image recognition in embedded systems using CNNs, which also reveals that computational and memory limitations should be considered, as shown in Table 3. The use of embedded systems has been studied by researchers using CNN architectures or models specifically designed to be deployed in embedded systems with limited capacity [34]. Moreover, the use of embedded systems such as minicomputers results in relatively fast inference times in medical applications [35], but it consumes more RAM. Another approach is to use a special random-access memory (RAM) architecture for a CNN to improve energy efficiency [36]. As an alternative, a field programmable gate array (FPGA) [37] has been used to perform specific classification tasks using CNNs with the aim of optimizing image augmentation and preprocessing data with hardware limitations. The inference time was significantly faster, and the memory usage was relatively small. However, the use of FPGA requires special programming skills.

TABLE 3
Comparison of CNN implementation on medical devices

Study/System	Application	Accuracy (%)	RAM Usage	Inference Time
Mohan et al. (2021) [34]	Face Mask Detection	99.79	496 KB	33 ms
Vincet et al. (2025) [35]	Skin cancer detection	98.25	1948 MB	10 ms
Chen et al. (2021) [36]	Medical Image Classification	98.8	8 KB	Peak Throughput is 573.4 GOPS (Giga Operations Per Second)
Ghani al. (2024) [37]	Kidney Cell Cytotoxicity Classification	91	8 BRAM from 225 KB	163.795 uS

Source: Own elaboration.

Currently, several microcontroller devices can be used for relatively uncomplex AI applications at relatively low cost. However, for relatively complex health applications that require high accuracy, minicomputers and FPGA, which are relatively expensive, are still required. Another critical factor is the availability of electricity in remote areas or whether the device is placed as a monitoring device. The solution that can be provided is to use solar cells or other renewable energy sources, such as wind. Therefore, a quantized model was developed to reduce the electricity consumption of the devices. Likewise, devices must be equipped with an intuitive interface, minimal maintenance and setup requirements, and a sturdy and robust casing to protect the device from various conditions. Devices should also be equipped with a manual in the local language so that it can be understood by users.

For future work, further research can be performed using advanced quantization techniques, such as mixed-precision [38], to improve accuracy with low memory usage. Another alternative is to use network pruning techniques [39] by eliminating redundant filters. Future research can also be conducted utilizing a more intricate multiclass classification, various NN architectures, or a moving object to determine whether embedded systems are capable of performing calculations with a higher level of complexity. In addition, real-world validation is needed for various applications in the medical field. This research can contribute to the creation of inexpensive, portable and intelligent medical equipment.

Conclusions

In this study, the application of DL for pneumonia detection was analyzed using X-ray images from an AI-based embedded system. Several input parameters, such as the number of layers, filters, kernels and input features, were varied to determine how these changes would affect the output parameters, such as accuracy, inference time, and RAM and flash usage. This was achieved by considering the limited memory and microprocessor capabilities of the embedded systems.

The results showed that, in some cases, an increase in filter number can increase accuracy, but this comes at the cost of increased inference time, RAM and flash usage. Therefore, it is necessary to focus on the correct configuration of the input parameters according to the capabilities of the embedded system without neglecting the output parameters. The use of the 32-bit unoptimized model also results in fairly large memory usage on embedded systems, so it is necessary to consider its use on embedded systems that have a small memory capacity.

In medical applications that require real-time and efficient decision-making, accuracy and inference time are critical. The two variables must be considered when selecting and determining a model. This is because medical applications are highly sensitive to various environmental conditions. Model optimization with quantization or other techniques is essential for creating a precise AI-based pneumonia detection tool for use in low-resource clinical environments.

Conflict of Interest

On behalf of all authors, the corresponding author states that there are no conflicts of interest.

References

[1] W. S. Lim, “Pneumonia—Overview,” *Reference Module in Biomedical Sciences*, pp. B978-0-12-801238-3.11636-8, 2020, doi: 10.1016/B978-0-12-801238-3.11636-8.

[2] V. Jain, R. Vashisht, G. Yilmaz, and A. Bhardwaj, “Pneumonia Pathology,” in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2022. Accessed: Dec. 20, 2022. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK526116/

[3] E. E. Adaji, W. Ekezie, M. Clifford, and R. Phalkey, “Understanding the effect of indoor air pollution on pneumonia in children under 5 in low- and middle-income countries: a systematic review of evidence,” *Environ Sci Pollut Res*, vol. 26, no. 4, pp. 3208–3225, Feb. 2019, doi: 10.1007/s11356-018-3769-1.

[4] L. Gattinoni *et al.*, “COVID-19 pneumonia: pathophysiology and management,” *European Respiratory Review*, vol. 30, no. 162, Dec. 2021, doi: 10.1183/16000617.0138-2021.

[5] A. Arora, “One is Too Many: Ending child deaths from pneumonia and diarrhoea,” UNICEF DATA. Accessed: Jan. 16, 2023. [Online]. Available: https://data.unicef.org/resources/one-many-ending-child-deaths-pneumonia-diarrhoea/

[6] P. Meenakshi., K. Bhavana, and A. K. Nair, “Pneumonia Detection using X-ray Image Analysis with Image Processing Techniques,” in *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, Jun. 2022, pp. 1657–1662. doi: 10.1109/ICCES54183.2022.9835798.

[7] C. Ortiz-Toro, A. García-Pedrero, M. Lillo-Saavedra, and C. Gonzalo-Martín, “Automatic detection of pneumonia in chest X-ray images using textural features,” *Comput Biol Med*, vol. 145, p. 105466, Jun. 2022, doi: 10.1016/j.compbiomed.2022.105466.

[8] I. Sirazitdinov, M. Kholiavchenko, T. Mustafaev, Y. Yixuan, R. Kuleev, and B. Ibragimov, “Deep neural network ensemble for pneumonia localization from a large-scale chest x-ray database,” *Computers & Electrical Engineering*, vol. 78, pp. 388–399, Sep. 2019, doi: 10.1016/j.compeleceng.2019.08.004.

[9] Z. Meng, L. Meng, and H. Tomiyama, “Pneumonia Diagnosis on Chest X-Rays with Machine Learning,” *Procedia Computer Science*, vol. 187, pp. 42–51, Jan. 2021, doi: 10.1016/j.procs.2021.04.032.

[10] H. Agrawal, “Pneumonia Detection Using Image Processing And Deep Learning,” in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, Mar. 2021, pp. 67–73. doi: 10.1109/ICAIS50930.2021.9395895.

[11] A. H. Alharbi and H. A. Hosni Mahmoud, “Pneumonia Transfer Learning Deep Learning Model from Segmented X-rays,” *Healthcare (Basel)*, vol. 10, no. 6, p. 987, May 2022, doi: 10.3390/healthcare10060987.

[12] R. Kundu, R. Das, Z. W. Geem, G.-T. Han, and R. Sarkar, “Pneumonia detection in chest X-ray images using an ensemble of deep learning models,” *PLOS ONE*, vol. 16, no. 9, p. e0256630, Sep. 2021, doi: 10.1371/journal.pone.0256630.

[13] T. Rajasenbagam, S. Jeyanthi, and J. A. Pandian, “Detection of pneumonia infection in lungs from chest X-ray images using deep convolutional neural network and content-based image retrieval techniques,” *J Ambient Intell Human Comput*, Mar. 2021, doi: 10.1007/s12652-021-03075-2.

[14] A. Mabrouk, R. P. Díaz Redondo, A. Dahou, M. Abd Elaziz, and M. Kayed, “Pneumonia Detection on Chest X-ray Images Using Ensemble of Deep Convolutional Neural Networks,” *Applied Sciences*, vol. 12, no. 13, Art. no. 13, Jan. 2022, doi: 10.3390/app12136448.

[15] A. K. Jaiswal, P. Tiwari, S. Kumar, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, “Identifying pneumonia in chest X-rays: A deep learning approach,” *Measurement*, vol. 145, pp. 511–518, Oct. 2019, doi: 10.1016/j.measurement.2019.05.076.

[16] W. Jeon, G. Ko, J. Lee, H. Lee, D. Ha, and W. W. Ro, “Chapter Six - Deep learning with GPUs,” in *Advances in Computers*, vol. 122, S. Kim and G. C. Deka, Eds., in Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, vol. 122. , Elsevier, 2021, pp. 167–215. doi: 10.1016/bs.adcom.2020.11.003.

[17] M. Pandey *et al.*, “The transformational role of GPU computing and deep learning in drug discovery,” *Nat Mach Intell*, vol. 4, no. 3, Art. no. 3, Mar. 2022, doi: 10.1038/s42256-022-00463-x.

[18] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria, “Machine Learning on Mainstream Microcontrollers,” *Sensors*, vol. 20, no. 9, Art. no. 9, Jan. 2020, doi: 10.3390/s20092638.

[19] J. Pardo, F. Zamora-Martínez, and P. Botella-Rocamora, “Online Learning Algorithm for Time Series Forecasting Suitable for Low Cost Wireless Sensor Networks Nodes,” *Sensors*, vol. 15, no. 4, Art. no. 4, Apr. 2015, doi: 10.3390/s150409277.

[20] J. Gorospe, R. Mulero, O. Arbelaitz, J. Muguerza, and M. Á. Antón, “A Generalization Performance Study Using Deep Learning Networks in Embedded Systems,” *Sensors*, vol. 21, no. 4, Art. no. 4, Jan. 2021, doi: 10.3390/s21041031.

[21] S. S. Saha, S. S. Sandha, and M. Srivastava, “Machine Learning for Microcontroller-Class Hardware: A Review,” *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21362–21390, Nov. 2022, doi: 10.1109/JSEN.2022.3210773.

[22] L. Saad Saoud and A. Khellaf, “A neural network based on an inexpensive eight-bit microcontroller,” *Neural Comput & Applic*, vol. 20, no. 3, pp. 329–334, Apr. 2011, doi: 10.1007/s00521-010-0377-5.

[23] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, “Memory-efficient Patch-based Inference for Tiny Deep Learning,” presented at the Annual Conference on Neural Information Processing Systems, Dec. 2021. Accessed: Dec. 20, 2022. [Online]. Available: https://research.ibm.com/publications/memory-efficient-patch-based-inference-for-tiny-deep-learning

[24] B. Almaslukh, “A Lightweight Deep Learning-Based Pneumonia Detection Approach for Energy-Efficient Medical Systems,” *Wireless Communications and Mobile Computing*, vol. 2021, p. e5556635, Apr. 2021, doi: 10.1155/2021/5556635.

[25] T. Barakbayeva and F. M. Demirci, “Fully automatic CNN design with inception and ResNet blocks,” *Neural Comput & Applic*, Sep. 2022, doi: 10.1007/s00521-022-07700-9.

[26] P. Kalgaonkar and M. El-Sharkawy, “CondenseNeXt: An Ultra-Efficient Deep Neural Network for Embedded Systems,” in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2021, pp. 0524–0528. doi: 10.1109/CCWC51732.2021.9375950.

[27] D. Menaka and S. G. Vaidyanathan, “Chromenet: a CNN architecture with comparison of optimizers for classification of human chromosome images,” *Multidim Syst Sign Process*, vol. 33, no. 3, pp. 747–768, Sep. 2022, doi: 10.1007/s11045-022-00819-x.

[28] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely Automated CNN Architecture Design Based on Blocks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020, doi: 10.1109/TNNLS.2019.2919608.

[29] M. Garifulla *et al.*, “A Case Study of Quantizing Convolutional Neural Networks for Fast Disease Diagnosis on Portable Medical Devices,” *Sensors*, vol. 22, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/s22010219.

[30] H. Liu, Q. Luo, M. Shao, J.-S. Pan, and J. Li, "Joint Channel Pruning and Quantization-Based CNN Network Learning with Mobile Computing-Based Image Recognition," *Wireless Communications and Mobile Computing*, vol. 2021, p. e9310558, Oct. 2021, doi: 10.1155/2021/9310558.

[31] J. Pérez, J. Díaz, J. Berrocal, R. López-Viana, and Á. González-Prieto, "Edge computing," Computing, vol. 104, no. 12, pp. 2711–2747, Dec. 2022, doi: 10.1007/s00607-022-01104-2.

[32] D. S. Kermany et al., "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," Cell, vol. 172, no. 5, pp. 1122-1131.e9, Feb. 2018, doi: 10.1016/j.cell.2018.02.010.

[33] L. J. Muhammad, A. A. Haruna, U. S. Sharif, and M. B. Mohammed, "CNN-LSTM deep learning based forecasting model for COVID-19 infection cases in Nigeria, South Africa and Botswana," Health Technol., vol. 12, no. 6, pp. 1259–1276, Nov. 2022, doi: 10.1007/s12553-022-00711-5.

[34] P. Mohan, A. J. Paul, and A. Chirania, "A Tiny CNN Architecture for Medical Face Mask Detection for Resource-Constrained Endpoints," in Innovations in Electrical and Electronic Engineering, S. Mekhilef, M. Favorskaya, R. K. Pandey, and R. N. Shaw, Eds., Singapore: Springer, 2021, pp. 657–670. doi: 10.1007/978-981-16-0749-3_52.

[35] Vincent, G. Darian, and N. Surantha, "Performance Evaluation of Convolutional Neural Network (CNN) for Skin Cancer Detection on Edge Computing Devices," Applied Sciences, vol. 15, no. 6, Art. no. 6, Jan. 2025, doi: 10.3390/app15063077.

[36] Z. Chen et al., "CAP-RAM: A Charge-Domain In-Memory Computing 6T-SRAM for Accurate and Precision-Programmable CNN Inference," IEEE Journal of Solid-State Circuits, vol. 56, no. 6, pp. 1924–1935, Jun. 2021, doi: 10.1109/JSSC.2021.3056447.

[37] A. Ghani, A. Aina, and C. Hwang See, "An Optimised CNN Hardware Accelerator Applicable to IoT End Nodes for Disruptive Healthcare," IoT, vol. 5, no. 4, Art. no. 4, Dec. 2024, doi: 10.3390/iot5040041.

[38] J. Wang, Z. He, H. Zhao, and R. Liu, "Low-Bit Mixed-Precision Quantization and Acceleration of CNN for FPGA Deployment," IEEE Transactions on Emerging Topics in Computational Intelligence, pp. 1–21, 2024, doi: 10.1109/TETCI.2024.3510295.

[39] E. Liberis and N. D. Lane, "Differentiable Neural Network Pruning to Enable Smart Applications on Microcontrollers," Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 6, no. 4, p. 171:1-171:19, Jan. 2023, doi: 10.1145/3569468.

Notes

* Research review

How to cite this article: Y. Abd Djawad, D. Mochamad Rifqie, Ridwansyah, Supriadi, S. Ronge Sokku, A. Baso Kaswar, M. Idris, "Performance Analysis of a Convolutional Neural Network for Pneumonia Detection on an Embedded AI System" Ing. Univ. vol. 29, 2025. https://doi.org/10.11144/Javeriana.iued29.panc