On-device Deep Learning for Recognizing 3D Geometric Shapes in an Educational App Using the TinyML Paradigm

André J A Ramos
Institute of Technology
Federal University of Pará
Belém-PA, Brazil
andre.rammos7@gmail.com

Roberto C. L. de Oliveira
Institute of Technology
Federal University of Pará
Belém-PA, Brazil
limao@ufpa.br

Manoel Campos Neto
Institute of Technology
Federal University of Pará
Belém-PA, Brazil
manoel.campos@itec.ufpa.br

***Abstract*—Currently, deep learning (DL) algorithms perform best in image classification and object detection tasks. Consequently, they are frequently used to address most problems involving computer vision. In this sense, the pervasive presence of smartphones and IoT devices has created a need to make this artificial intelligence portable. Given that deep neural network (DNN) models consist of millions of parameters, emerging research efforts have focused on enabling offline DL execution on low-resource devices, such as within the TinyML paradigm. This study analyzes the state-of-the-art of DL embedded in smartphones to develop an app for children that can recognize 3D geometric shapes without needing an internet connection. Alongside a systematic literature review, we conduct experiments with several pre-trained and lightweight models, which were subsequently evaluated using parametric statistical tests. While DL on smartphones is an underexplored area, it is expected to evolve significantly. Among the classification models tested, DenseNet169 demonstrated the highest accuracy (81%), whereas the MobileNet variants were faster and closer to real-time performance (30 FPS). In detection tasks, the EfficientDet-Lite and YOLOv8 models were evaluated, with EfficientDet-Lite being less accurate but faster (50 ms) compared to YOLOv8 (4 seconds). Although the field of DL on smartphones still requires further development, current lightweight models and frameworks offer significant opportunities for practical application.**

***Index Terms*—computer vision, deep learning, education, tinyml**

I. Introduction

ALGORITHMS for classifying and detecting objects in images have advanced significantly with the development of deep learning (DL) models. Numerous models have been created and frequently reported in the computer science and engineering literature. These models are designed to meet the requirements of various applications while operating on different devices, including microcontrollers (MCUs) and smartphones, offering high precision with minimal memory, storage, processing and energy consumption.

These limitations present a significant bottleneck in the engineering of these architectures, particularly with convolutional neural networks (CNNs), which consist of sequences of layers containing millions of parameters. For instance, pioneering models such as AlexNet contain 60 million parameters, demanding substantial memory, processing power (720 million FLOPs) and energy [12].

Cloud computing has already been proposed as a solution to this problem, as running these models on large, powerful machines with abundant resources typically presents no problems. However, when it comes to smaller devices, engineers often face a trade-off between two crucial factors in model development: accuracy and inference speed [7].

In this context, [38] points out that with the future increase in the number of Internet of Things (IoT) devices, the cloud may become unable to handle the vast amount of data generated. In 2020, [7] estimated that there were nearly 7 billion IoT devices and 3 billion smartphones worldwide, and by this year (2025), they predicted there would be 8.9 billion of them. As a result, there is a trend to shift server-side processing, traditionally performed in data centers, to end devices closer to the user or data source, including smartphones [8].

Additionally, the on-device DL approach is a crucial requirement for smart applications that must operate without an internet connection, such as medical applications [24], autonomous vehicles [42], drone rescue missions [20], aerial inspection of power transmission lines [44], and smartphones outside mobile networks, among others. Consequently, DL has numerous applications, and when it comes to processing images on smartphones (including offline), many other use cases have emerged, several of which are already documented in the literature. Examples include aiding decision-making in medical imaging diagnosis [30], visual identification of diseases in agriculture [35], counting fruits on trees during large harvests [47], and augmented reality [15]. Many of these examples adopt the on-device approach or plan to do so in future work, though some still rely on cloud-based inference in their experiments.

In the context of education, which is the focus of this study, deep learning (DL) combined with Virtual, Augmented, and Mixed Reality (VR, AR, and MR) helps create various educational tools, such as those that use VR headsets for teaching [32]. Many AI and DL applications rely on the cloud for data processing; however, connectivity remains limited in many regions of Brazil and across the globe.

According to data from the National Telecommunications Agency (Anatel), the number of schools without internet access in Brazil has been decreasing over the years, but it still affects 5.5% of schools, which corresponds to 7,592 institutions, leaving approximately 368,519 students without internet access [9]. Embedding DL in mobile devices allows intelligent processing, such as image recognition, to be performed locally, eliminating the need for a connection to a remote server. A research project focused on transforming this AI into mobile applications for early childhood education has been carried out by inteceleri[1], a company headquartered at the Guamá Science and Technology Park in Belém, Pará, Brazil, which develops technologies for mathematics education for early-grade students.

The company offers the Geometa (for Android[2] and IOS[3]) application with the motto 'Learn geometry through the Metaverse.' The app aims to make learning geometry more engaging and educational for students within a Metaverse environment, with a planned future feature to incorporate the recognition of 3D geometric shapes in images captured in real-time by the camera. Therefore, this research aims to move intelligent processing from the cloud to the device. Our main hypothesis is that the tinyML paradigm and its techniques, such as quantization, are sufficiently useful to optimize deep learning models applied to object detection with geometric shapes.

Specifically, this study provides the following contributions:

1) A brief analysis of the nine main lightweight DL models used for classifying and detecting 3D geometric shapes on smartphones.
2) A compilation of the most frequent techniques and frameworks and optimization measures for deep learning models employed by researchers in the literature, obtained through a systematic review.
3) A robust detection dataset [4] labeled with the four main 3D geometric shapes.

II. Background

A. Deep Learning

Deep learning is a branch of machine learning. In this context, depth refers to the number of trainable neuron layers, which increases the network's hypothesis space. According to [38], a deep neural network (DNN) is considered deep if it consists of at least three hidden layers.

a) Convolutional Neural Networks (CNNs): The network's convolutional layers can automatically extract features from the input, a task that, in traditional ML, required human expertise, enabling the model to understand complex patterns and achieve superior performance in most classification tasks [41]. While convolutional layers reduce the number of parameters compared to DNNs with fully connected layers, they demand more computation, accounting for approximately 90% of the network's processing [7].

b) Transfer Leaning: To achieve satisfactory results, CNNs require training with massive amounts of data, making it challenging to gather sufficient images for specific domains. This is where the benefit of transfer learning comes in. Transfer learning (TL) is an efficient and widely used approach for building DL models, as it facilitates the training process even with limited data by leveraging pre-trained weights or features from more extensive datasets, such as COCO[5] or ImageNet[6]. This approach enables the application of learned knowledge to new, smaller datasets [41].

B. TinyML

Research on TinyML under this name began in 2019 [33]. The author in [34] defines it as a paradigm that simplifies the integration of artificial intelligence (AI) algorithms into embedded devices, requiring significantly fewer processing, energy, and memory resources. Systems equipped with this technology can make decisions locally, within the physical scope of embedded devices, before connecting to AI at the edge or in the cloud, thereby minimizing connectivity needs while ensuring energy efficiency, privacy, and low latency.

According to a survey conducted by ABI Research, it is estimated that by 2030, around 2.5 billion small devices implementing TinyML techniques will enter the market [2]. This growth is being driven by the increasing number of hardware and software platforms developed for research and compatible with the TinyML paradigm [34].

According to [33], TinyML introduces several optimization techniques applicable to DL models during the training phase, the most commonly used being pruning, knowledge distillation (KD) and quantization.

a) Pruning: As the name suggests, this is a technique for eliminating connections between neurons, weights, filters or layers that have little influence on the accuracy of a trained model. Since DL models are overparameterized and have a large hypothesis space, this ends up allowing some weights to emerge with little influence on the general calculation performed by the network [7], [38]. According to [7], the process of pruning these parameters was originally introduced by LeCun et al. (1990) and one of its benefits is to reduce the size occupied by the network in memory and to speed up the inference time, since it reduces the number of multiplication and accumulation (MAC) operations performed by the convolution of the filters, in the case of a CNN. Pruning also benefits the model by reducing overfitting during adjustment, considering that some connections can contribute more or less to the reduction of the network's error function [19]. Thus, [7] also comments that the act of pruning is a way of searching for the best architecture for a neural network.

b) Knowledge distillation (KD): According to [7], this method was introduced by Buciluǎ et al. in 2006 and is

[1]https://www.inteceleri.com.br

[2]https://play.google.com/store/apps/details?id=com.Inteceleri.Geometa

[3]https://apps.apple.com/br/app/geometa/id1644556020

[4]https://universe.roboflow.com/geomagia/geometria

[5]https://cocodataset.org/

[6]https://www.image-net.org/

useful when there is a need to develop models that run in environments that operate with low computational resources. The technique consists of transferring the knowledge learned by a more complex network (called the ”teacher model”) to a smaller and lighter network (called the ”student model”). However, this technique is still limited to classification models and, until the publication of [7], KD was considered a challenge to implement for detection and segmentation models.

c) Quantization: This technique adds even more to the optimization of models, reducing the numerical precision of weights and activations in smaller bits (16 bits, 8 bits, 4 bits) without changing the network architecture, resulting in a reduction in storage and computational load. In some cases, quantization benefits models, making them faster and lighter [19]. According to [7], quantization has its origins in the 1990s, in [14] and [4] and was designed to facilitate the implementation of artificial neural networks (ANN) in hardware.

III. Methodology

This research is guided by the following question: What is the most effective DL approach, tool, and algorithm to embed in mobile communication devices for classifying and detecting objects in images?

To address this question, we conducted a systematic literature review (SLR) alongside experiments on smartphones, the target devices of this study, to achieve the primary objective of this research: improving DL models and embedding them in these mobile devices with limited resources.

A. Dataset construction

a) Classification: The set of images used to train the classification models was a variation of the dataset proposed by [10], based on ObjectNet[7], which we expanded with additional images captured by smartphones and others extracted from public Roboflow[8] directories (see Figure 1). The final dataset contained a significant number of images with good sharpness and illumination, closely resembling captures made by smartphone cameras.

Furthermore, the dataset was multiplied and enriched through data augmentation (see Table I) performed using the Roboflow service. As a result, the number of images increased from 5,693 to 14,686.

In the modifications, only a few degrees of rotation were used (90º and 45º), and shearing was strategically avoided so as not to distort the geometric shape of the objects and, consequently, interfere with the prediction results. The modifications made to the images were:

1) Inverted: Horizontal, Vertical
2) 90º rotation: clockwise, counterclockwise, upside down
3) Rotation: between -45° and +45°
4) Brightness: between -15% and +15%
5) Noise: above 1.8% of pixels

[7]https://objectnet.dev/download.html
[8]https://roboflow.com/

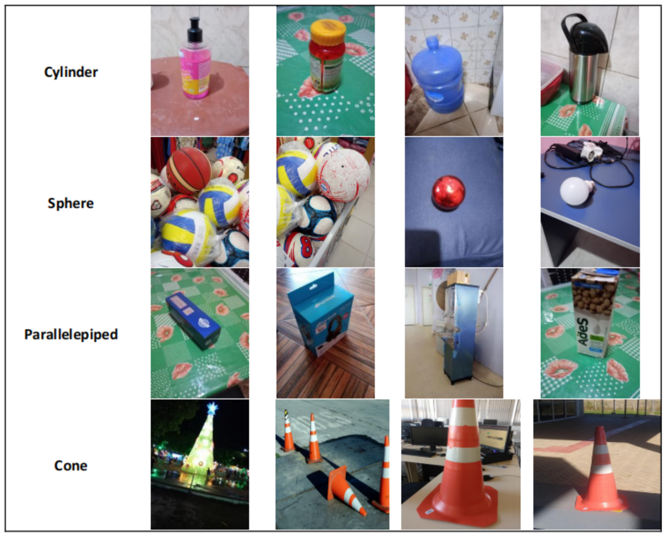

Fig. 1. Sample images of each type of geometric shape present in the dataset.

TABLE I
Number of images of each class and section in the classification dataset.

	Training	*Validation*	*Test*	*Total*
Cylinder	891	125	120	1,136
Cone	561	125	92	778
Sphere	813	100	99	1,012
Undefined	805	112	112	1,029
Parallelepiped	807	113	89	1,009
Surface	580	74	75	729
Total	4,457	649	587	5,693
Percentage	78%	11%	11%	100%

b) Detection: The dataset used for the detection task in this study was partially composed of images obtained via smartphone and partially sourced from publicly available databases of other projects on Roboflow. The dataset was further expanded through data augmentation (see Table II). As you know, the augmentation process multiplies only the training data, from 2,845 to 8,535, giving a total of 9,641 images of size 840x840.

TABLE II
Number of images of each class and in each section in the detection dataset (before the augmentation process).

	Training	*Validation*	*Test*	*Total*
Cylinder	1,065	491	395	1,951
Cone	1,414	394	62	1,870
Sphere	1,152	331	160	1,643
Parallelepiped	1,500	232	118	1,850
Total	5,131	1,448	735	7,314
percentage	70%	20%	10%	100%

B. Experiment details

The models were subjected to multiple training sessions with the constructed datasets to statistically validate the performance comparison between them. We conducted ten training sessions for each model using different random seeds. To determine the appropriate statistical method, we analyzed the results from all training sessions using the Shapiro-Wilk test [40], which confirmed a normal distribution of the data. Consequently, the parametric ANOVA (Analysis of Variance) method was selected for the analysis [22], as it is considered the most common way to evaluate the difference between the means of three or more groups [26].

While the analysis of variance determines whether or not there is a global difference in the groups, a specific test for each pair becomes necessary to verify whether there are significant differences between them, and these tests are known as post-hoc [25]. According to [25], the Tukey test is the most preferable method for this. Several free or non-free software programs provide this analysis in full, such as the Scipy[9] library itself from Python and the Jamovi® software [43].

We carried out experiments using nine small, lightweight, and quantized TensorFlow models on the 3D geometric shapes dataset, covering both classification tasks (Xception, InceptionV3, MobileNet, MobileNetV2, DenseNet169, NASNetMobile, EfficientNetV2B0) and detection tasks (YOLOv8 and EfficientDet models). All these nine models were pre-trained on more robust datasets through the TL process: classification models were pre-trained with the ImageNet dataset, while detection models used the COCO dataset. The EfficientDetLite detection models were compared with a YOLOv8 network. It is important to note that the goal of these experiments was not to identify a model that outperforms all others across various tasks or datasets. As [18] rightly points out, the 'No Free Lunch' theorem asserts that no single algorithm performs optimally across all datasets, but certain algorithms tend to yield better results for specific datasets.

The steps for creating and deploying the DL model on smartphones are as follows:

1) Create and train the models using Keras in a notebook (.ipynb) running on a personal computer (PC).
2) Save the model in the .keras format and convert it to the FlatBuffer format with the extension .tflite.
3) Optimize the models with quantization that reduces the .tflite file even further.
4) Add the .tflite file to the project assets in Android Studio.
5) Install the built package on the smartphone.
6) The smartphone processes images from the camera and makes predictions in real-time.

IV. Literature Overview

Based on the terminology obtained in the exploratory analysis process, a search string was created, which in its raw form was as follows: ((”Deep learning” OR ”Machine learning”)

[9]https://scipy.org

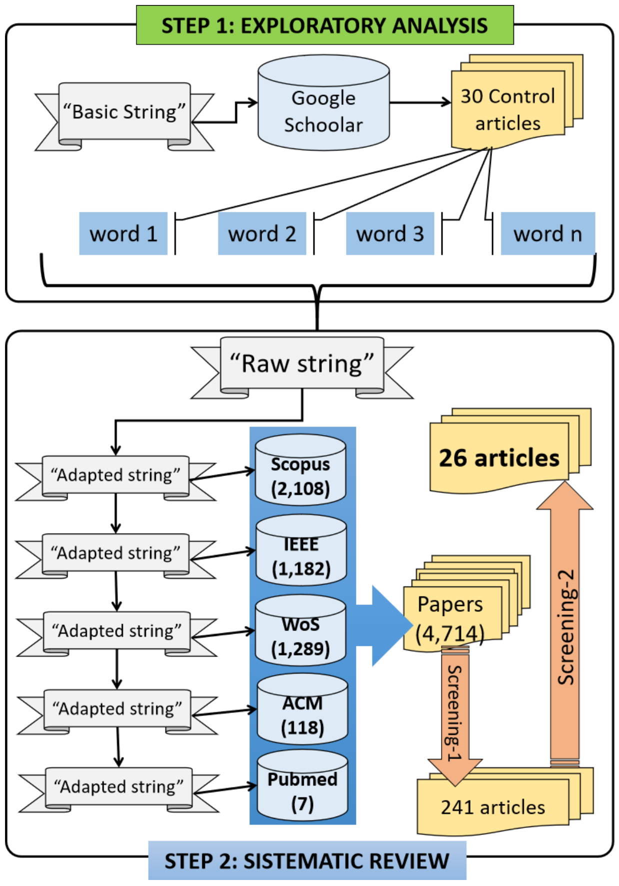

Fig. 2. Summary of the phases employed in the RSL, as well as the process of constructing the search strings submitted to the databases. The term screening-1 refers to the process of reading abstracts and screening-2 to the reading of the articles in full.

AND (TinyML OR Tiny-ML OR ”Lightweight” OR ”constrained computing resources” OR ”Resource-constrained” OR ”Resource Constraints”) AND (smartphone OR ”smartphone-based” OR ”mobile device” OR ”Mobile phones” OR ”Android” OR ”IOS” OR ”Iphone” OR ”Mobile Application”) AND (”On-device DL” OR ”On-device Deep Leaning” OR ”Embedded Machine Learning”) AND (”Image classification” OR ”object detection” OR ”image detector” OR image)). The databases selected for the search were: IEEE, Pubmed, Scopus, ACM and Web of Science.

Thus, applying the search string cited and adapting it to the bibliographic databases cited, 4,714 articles were returned in total. With duplicates removed, 3,631 articles remained, which underwent the abstract reading process (screening), filtering them using pre-established inclusion and exclusion criteria, resulting in 241 articles (see Figure 2).

When read in full, these 241 articles were subjected to the same inclusion/exclusion filters and, in this way, these criteria were highlighted in all of them, leaving only the 26 articles.

Through Figure 3, it is possible to have a glimpse of which countries publish the most on the subject in question, originating from the search with the strings submitted to the databases according to the results presented in the Scopus

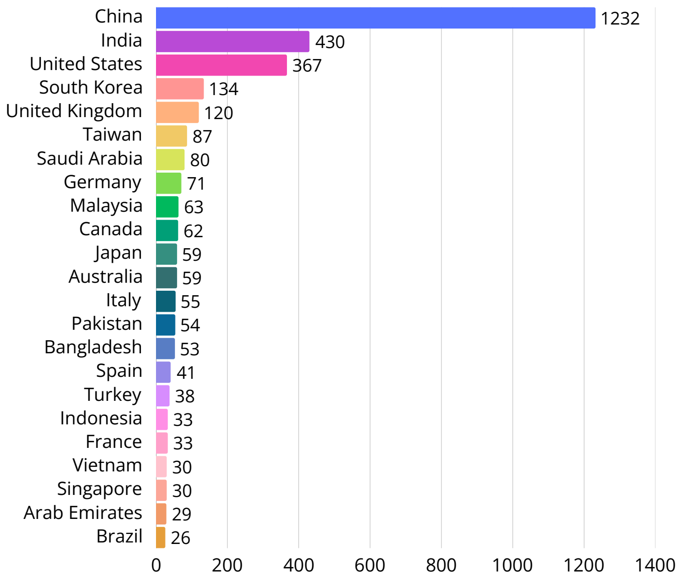

Fig. 3. Graph of the number of articles returned with the theme from the search string applied in the Scopus database in different countries between the years 2015 and 2024.

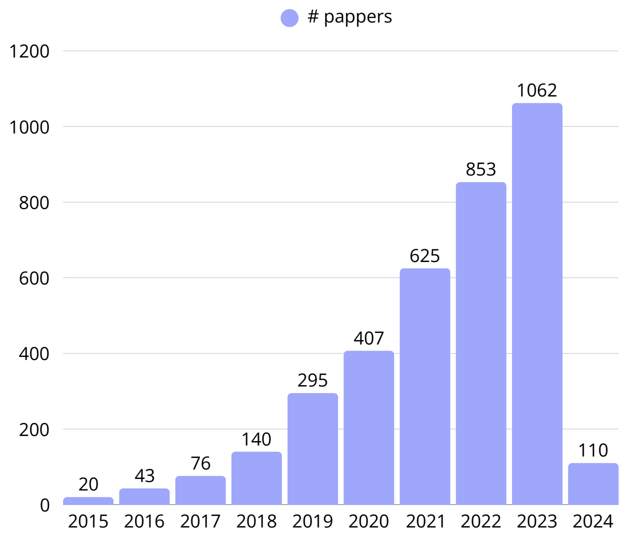

Fig. 4. Annual distribution of articles in a bar graph showing the evolution of interest in research over 8 years (Jan. 2015–Mar. 2024).

database. China is the country with a number of published articles almost three times greater (1,232) than the second-placed India (430). In Kai-Fu Lee's analysis [27], the reason behind this is the amount of data produced, collected and processed by the country with its large population, suggesting that the monopoly of AI can be achieved by the nation that produces and exploits the most data.

With the articles pointed out by the bibliographic databases, a significant growth in interest in the subject of deep learning on mobile devices was also observed over the last 8 years (Jan. 2015–Mar. 2024), which can be seen in the two graphs shown in Figure 4.

The Rayyan software[10], used in this work, automatically classified the types of publication on the initial amount of 3631 articles returned as being 70% (2,536) journal articles and 30% (1,094) conference articles.

The work by [36] attributes the accelerated development of DL to the availability of public, free libraries. A study by [11] identified TensorFlow Lite as the most widely used tool for on-device DL inference. In March 2021, [46], using a web crawler, identified 676 DL applications in the Play Store, finding that the most commonly used DL libraries were TensorFlow Lite (70.5%), followed by TensorFlow (7.8%), ncnn (7.2%), Caffe (4.4%), MNN (4.1%), PyTorch Mobile (3.8%), and Mace (1.2%).

An analysis of the most relevant system review papers showed that the researchers made effective use of frameworks made available by big tech companies, with TensorFlow Lite (by Google) being the most prevalent (13) adopted by the articles for the task of optimizing DL models for devices with limited resources (see Table III). Our work is part of this group. While three other authors adopted pure TensorFlow, another adopted Core ML (by Apple)[11], and another group of colleagues adopted more generic frameworks such as OpenCV[12] and Edge Impulse[13] .

To measure the efficiency of the models used, modified or proposed by researchers, it was found that not all adopted the same measure. However, most studies (including ours) measured the number of parameters and the inference time.Others evaluated RAM consumption, device storage usage, and some measured the number of FLOPs. Our study included all these metrics (see Table III).

The review also revealed that, although most authors tested their models on smartphones, not all actually deployed lightweight models on such constrained devices. A small number of key studies still performed their tests on a Personal Computer (PC). Our study conducted experiments on both a PC and a smartphone (see Table III).

In terms of measuring the accuracy of the models tested, the studies adopted different measures according to their purposes. As can be seen in Table III, the measures used were: precision, average precision (AP), accuracy, F1-score, recall, and confusion matrix. Most of the studies found used a combination of precision, recall and F1-score. Our study employed all of them.

The work by [21] conducted a SLR aimed at identifying trends in the use of DL on resource-constrained devices, concluding that, in terms of academic research, DL on mobile devices still has a significant gap requiring further studies. One of them refers to the lack of quantitative data not exposed in articles found so that a meta-analysis could be done, for example, something that could add a lot to SLR work. In 2021, [3] noted that although some of the mentioned tools enable on-device inference, they are still far from being universally

[10]https://new.rayyan.ai
[11]https://developer.apple.com/documentation/coreml
[12]https://opencv.org/
[13]https://edgeimpulse.com

TABLE III
LIST OF WORKS AND THEIR APPROACHES

Features	Papers
Tiny method	
Quantization	[47], [30], [41], [1], [29], [37], [Our work]
LeakyRelu, Map dilation	[17]
ShuffleNet	[13]
CSP	[5], [39], [31], [45]
Ligth models SOTA	[6], [16], [23], [35]
Devices	
PC	[39], [31], [37], [45], [17], [13], [Our work]
Smartphone	[5], [47], [35], [30], [41], [1], [28], [29], [6], [16], [our work]
Raspberry	[23]
Measure performance models	
Precision	[30], [41], [13], [28], [17], [31], [6], [16], [37], [35], [Our work]
Recall	[29], [31], [6], [16], [35], [37], [Our work]
F1-score	[23], [29], [17], [31], [6], [16], [35], [Our work]
Accuracy	[39], [45], [17], [37], [31], [6], [16], [Our work]
Confusion matrix	[35], [Our work]
Average precision (AP)	[29], [23], [Our work]
Efficient performance on device	
Nº parameters	[30], [39], [5], [45], [31], [Our work]
Memory	[1], [13], [28], [5], [Our work]
Model size	[39], [45], [Our work]
FLOPs	[5], [45], [31], [Our work]
Frameworks used	
TensorFlow Lite	[5], [47], [35], [30], [41], [1], [13], [28], [29], [16], [23], [Our work]
TensorFlow	[39], [31], [17]
CoreML	[6]
Others	[28], [37]

applicable, as only 50% of devices currently possess the necessary hardware resources for execution—an important factor to consider.

According to [33], the Kotlin, PyTorch and Keras Application Programming Interface (APIs) for DNNs use C++ 'under the hood,' as they operate in a layer closer to the hardware, which speeds up data processing. The work of [33] identifies smartphones as intermediate devices positioned between edge computing and cloud computing paradigms.

V. EXPERIMENTS

A. Setup

For the training stage, we use a computer with a Linux Ubuntu operating system (OS), equipped with an NVIDIA GeForce RTX 3060 graphics card (12 GB GDDR6 memory), 1 GPU, and an AMD Ryzen 7 5700X CPU(3.4 GHz, 8 cores, 16 threads), along with 16GB of RAM. The system operated in a Python 3.11 and CUDA 11.5 environment. For the inference stage of the experiment on the smartphone, we used a physical Motorola Moto g23 device with an ARM Cortex-A75 processor (2.00 GHz) and Android OS version 14.

B. Results

After tuning, optimizing, and evaluating the models, the best-performing ones on the PC were embedded in an Android mobile application to measure inference time and monitor RAM consumption. The app was developed using the TensorFlow Lite API in Kotlin language and is available for download on the Play Store under the name Geomagia[14] for public use and experimentation (see Figure 5).

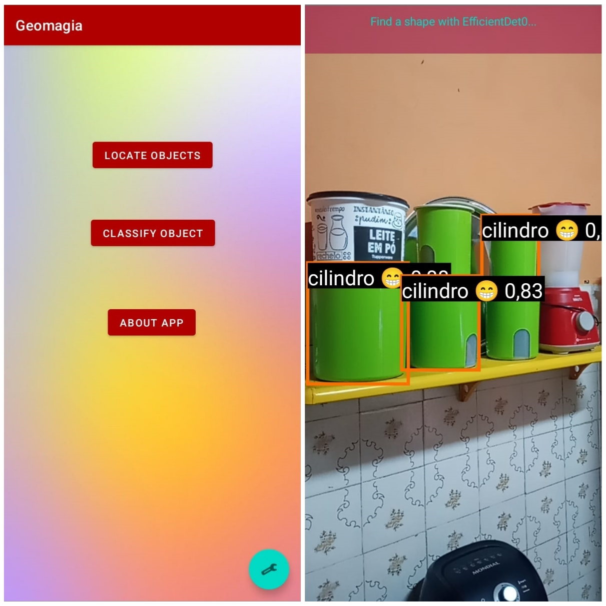

Fig. 5. Screenshots of the Android application Geomagia detecting objects which was developed to test the performance of the models on the smartphone.

Regarding the classification models, Table IV presents the results for accuracy, RAM consumption, frames per second (FPS), and average inference time (AIT) measured on the device. The Table V shows the F1-score for each class in the dataset. Fig. 6 displays, through a confusion matrix, the accuracy of the winning model (DeseNet169) for each of the classes. Experiments with the detection models were also conducted, with EfficientDetLite being faster (50 ms) but less accurate ($mAP \approx 32\%$), while YOLOv8 was more accurate ($mAP50 = 95\%$) but significantly slower (4 seconds).

As shown in Table IV, we can see that the mobileNetV2 model and its quantized version consume much less RAM (134.87MB and 125.71MB, respectively) and have a shorter average inference time than all the other models. On the other hand, its accuracy is only 74%. While the most accurate model is DenseNet169 and its quantized version reaches 81% and 82%, respectively, it consumes a lot of RAM (over 200MB),

[14]https://play.google.com/store/apps/details?id=com.edusmart.geomagia

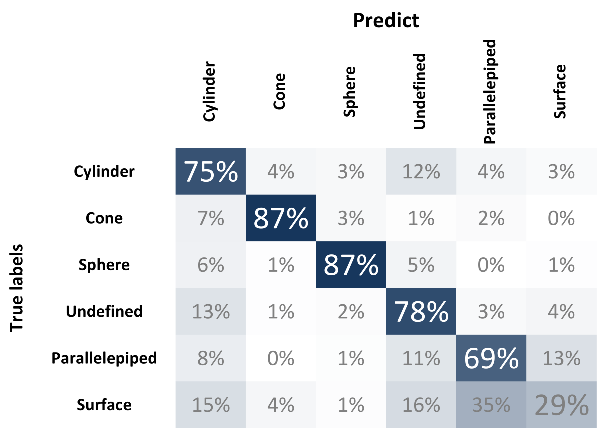

Fig. 6. Confusion matrix showing the accuracy of the winning model (DeseNet169) for each of the classes.

showing the inevitable trade-off between the accuracy and size of these models.

TABLE IV
PERFORMANCE OF DEEP LEARNING MODELS TESTED ON SMARTPHONE

Model	RAM(MB)	AIT(ms)	FPS	Accuracy(%)
MobileNetV2	**134.87**	**40.87**	**24.47**	74%
MobileNetV2_qtz	**125.71**	**51.21**	19.53	74%
MobileNet_qtz	132.48	54.47	18.36	74%
MobileNet	144.75	60.82	16.44	74%
NASNetMobile_qtz	134.19	105.58	9.47	72%
NASNetMobile	175.74	125.99	7.94	74%
EfficientNet_qtz	120.08	139.02	7.19	54%
InceptionV3-qtz	141.57	158.51	6.31	72%
EfficientNet	155.24	172.81	5.79	77%
InceptionV3	284.22	240.78	4.15	72%
Xception_qtz	161.12	265.29	3.77	79%
DenseNet169_qtz	200.37	268.28	3.73	**82%**
DenseNet169	227.62	323.79	3.09	**81%**
Xception	297.99	416.38	2.40	79%

F1-score measure was selected because it takes into account both precision and recall, since both are important for critical applications like ours that do not tolerate false positives or false negatives. Once again, we see the high efficiency values achieved by the densest models, such as DenseNet169 which was the best of all the others, with mobileNet achieving the lowest values because, in theory, it is a small model. We also observed a general difficulty in the models recognizing some objects, such as surfaces.

VI. Conclusion

The studies have shown that quantization is a widely used technique for making DL models lightweight. However, the diversity of solutions—expressed through different architectures, software, libraries, and languages—poses challenges in establishing a common standard across all applications and devices, including smartphones.

TABLE V
PERFORMANCE OF THE SEVEN CLASSIFICATION MODELS ACCORDING TO THE F1-SCORE MEASURE.

	DenseNet169	*EfficientNetV2B0*	*InceptionV3*	*MobileNet*	*MobileNetV2*	*NASNetMobile*	*Xception*
Cylinder	***78%***	*73%*	*71%*	*69%*	*69%*	*68%*	*76%*
Cone	*93%*	*93%*	*88%*	*90%*	*89%*	*91%*	***94%***
Sphere	***95%***	*90%*	*88%*	*90%*	*89%*	*90%*	*92%*
Undefined	***80%***	*77%*	*72%*	*71%*	*72%*	*71%*	*76%*
Parallelepiped	***74%***	*67%*	*65%*	*67%*	*64%*	*63%*	*72%*
Surface	***63%***	*59%*	*37%*	*49%*	*49%*	*52%*	*63%*
macro avg	***81%***	*77%*	*70%*	*73%*	*72%*	*73%*	*79%*
weighted avg	***81%***	*77%*	*71%*	*73%*	*73%*	*73%*	*79%*

Our research and experiments demonstrate that significant progress can be made by leveraging existing technologies for mobile devices, as the efficiency and accuracy achieved by established networks are evident for the proposed educational application of this work.

The use of AR in early childhood education, especially for the identification of geometric shapes, presents itself as an innovative tool with strong pedagogical potential. However, it is essential to consider its technical, physical, and pedagogical limitations. The success of the application depends on the quality of the device, teacher mediation, and integration with other more traditional teaching strategies, such as concrete games and hands-on activities.

In future work, the proposed architecture or a variation of an existing one could be explored for the classification or detection of geometric objects, incorporating the optimization techniques discussed in this article. Researchers interested in integrating the practicality of the FlatBuffers API, which interprets TensorFlow Lite models, into the Unity3D game environment, could consider developing these interpreters in C# language for use in AR and VR applications.

Acknowledgment

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil* (CAPES).

References

[1] J. Ahmad, B. Jan, H. Farman, W. Ahmad, and A. Ullah, "Disease detection in plum using convolutional neural network under true field conditions," *Sensors*, vol. 20, no. 19, p. 5569, Sep. 2020.
[2] N. N. Alajlan and D. M. Ibrahim, "Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications," *Micromachines*, vol. 13, no. 6, p. 851, May 2022.
[3] Y. Bai, L. Chen, M. Abdel-Mottaleb, and J. Xu, "Automated ensemble for deep learning inference on edge computing platforms," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4202–4213, Mar. 2022.
[4] W. Balzer, M. Takahashi, J. Ohta, and K. Kyuma, "Weight quantization in boltzmann machines," *Neural Networks*, vol. 4, no. 3, pp. 405–409, Jan. 1991.

[5] S. Biswas and S. Barma, "Microsmobinet: A deep lightweight network with hierarchical feature fusion scheme for microscopy image analysis in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 8288–8298, Mar. 2024.

[6] Y.-C. Chen, Y.-C. Chu, C.-Y. Huang, Y.-T. Lee, W.-Y. Lee, C.-Y. Hsu, A. C. Yang, W.-H. Liao, and Y.-F. Cheng, "Smartphone-based artificial intelligence using a transfer learning algorithm for the detection and diagnosis of middle ear diseases: A retrospective deep learning study," *eClinicalMedicine*, vol. 51, p. 101543, Sep. 2022.

[7] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev.*, vol. 53, no. 7, pp. 5113–5155, Feb. 2020.

[8] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog," *IEEE Access*, vol. 7, pp. 150 936–150 948, 2019.

[9] A. N. de Telecomunicações. (2024) Anatel - conectividade nas escolas. Accessed on August 16, 2024. [Online]. Available: https://informacoes.anatel.gov.br/paineis/infraestrutura/conectividade-nas-escolas

[10] A. M. Dos Santos, F. R. T. Moura, A. V. N. Alves, L. V. L. Pinto, F. A. R. Costa, W. Dos Santos Oliveira Júnior, D. L. Cardoso, and M. C. Da Rocha Seruffo, "Artificial intelligence in education 5.0: a methodology for three-dimensional geometric shape classification for an educational tool," in *2023 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, Oct. 2023.

[11] O. Durmaz Incel and S. O. Bursa, "On-device deep learning for mobile and wearable sensing applications: A review," *IEEE Sens. J.*, vol. 23, no. 6, pp. 5501–5512, Mar. 2023.

[12] O. Elharrouss, Y. Akbari, N. Almadeed, and S. Al-Maadeed, "Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision," *Computer Science Review*, vol. 53, p. 100645, Aug. 2024.

[13] G. Fan, F. Chen, D. Chen, Y. Li, and Y. Dong, "A deep learning model for quick and accurate rock recognition with smartphones," *Mobile Information Systems*, vol. 2020, pp. 1–14, May 2020.

[14] E. Fiesler, A. Choudry, and H. J. Caulfield, "Weight discretization paradigm for optical neural networks," in *Optical Interconnections and Networks*, H. Bartelt, Ed., vol. 1281. SPIE, Aug. 1990, p. 164.

[15] Y. Ghasemi, H. Jeong, S. H. Choi, K.-B. Park, and J. Y. Lee, "Deep learning-based object detection in augmented reality: A systematic review," *Comput. Ind.*, vol. 139, p. 103661, Aug. 2022.

[16] J. Glory Precious, S. P. Angeline Kirubha, and I. Keren Evangeline, "Deployment of a mobile application using a novel deep neural network and advanced pre-trained models for the identification of brain tumours," *IETE Journal of Research*, vol. 69, no. 10, pp. 6902–6914, Jun. 2022.

[17] E. Goceri, "Diagnosis of skin diseases in the era of deep learning and mobile technology," *Computers in Biology and Medicine*, vol. 134, p. 104458, Jul. 2021.

[18] M. Harrison, *Machine Learning Pocket Reference: Working with Structured Data in Python*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, Sep. 2019.

[19] S. Hong, G. Park, and J. Kim, "Automated deep-learning model optimization framework for microcontrollers," *ETRI Journal*, Jun. 2024.

[20] K. Jayalath and S. R. Munasinghe, "Drone-based autonomous human identification for search and rescue missions in real-time," in *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, Aug. 2021, pp. 518–523.

[21] V. Kamath and A. Renuka, "Deep learning based object detection for resource constrained devices: Systematic review, future trends and challenges ahead," *Neurocomputing*, vol. 531, pp. 34–60, Apr. 2023.

[22] L. S. Kao and C. E. Green, "Analysis of variance: Is there a difference in means and what does it mean?" *Journal of Surgical Research*, vol. 144, no. 1, pp. 158–170, Jan. 2008.

[23] E. Kee, J. J. Chong, Z. J. Choong, and M. Lau, "A comparative analysis of cross-validation techniques for a smart and lean pick-and-place solution with deep learning," *Electronics*, vol. 12, no. 11, p. 2371, May 2023.

[24] A. R. Keivanimehr and M. Akbari, "Tinyml and edge intelligence applications in cardiovascular disease: A survey," *Computers in Biology and Medicine*, vol. 186, p. 109653, Mar. 2025.

[25] H.-Y. Kim, "Statistical notes for clinical researchers:post-hoc multiple comparisons," *Restorative Dentistry; Endodontics*, vol. 40, no. 2, p. 172, 2015.

[26] T. K. Kim, "Understanding one-way anova using conceptual figures," *Korean Journal of Anesthesiology*, vol. 70, no. 1, p. 22, 2017.

[27] K. Lee and M. Barbão, *Inteligência artificial*. Globo Livros, 2019. [Online]. Available: https://books.google.com.br/books?id=0zO7DwAAQBAJ

[28] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Smartphone-based real-time object recognition architecture for portable and constrained systems," *Journal of Real-Time Image Processing*, vol. 19, no. 1, pp. 103–115, Sep. 2021.

[29] R. Nakasi, E. Mwebaze, and A. Zawedde, "Mobile-aware deep learning algorithms for malaria parasites and white blood cells localization in thick blood smears," *Algorithms*, vol. 14, no. 1, p. 17, Jan. 2021.

[30] D. Neogi, M. A. Chowdhury, M. M. Akter, and M. I. A. Hossain, "Mobile detection of cataracts with an optimised lightweight deep edge intelligent technique," *IET Cyber-Physical Systems: Theory; Applications*, Jan. 2024.

[31] C. Ntakolia, D. E. Diamantis, N. Papandrianos, S. Moustakidis, and E. I. Papageorgiou, "A lightweight convolutional neural network architecture applied for bone metastasis classification in nuclear medicine: A case study on prostate cancer patients," *Healthcare*, vol. 8, no. 4, p. 493, Nov. 2020.

[32] A. Pregowska, M. Osial, D. Dolega-Dolegowski, R. Kolecki, and K. Proniewska, "Information and communication technologies combined with mixed reality as supporting tools in medical education," *Electronics*, vol. 11, no. 22, p. 3778, Nov. 2022.

[33] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable tinyml," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–30, Jul. 2023.

[34] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 4, pp. 1595–1623, Apr. 2022.

[35] M. Reda, R. Suwwan, S. Alkafri, Y. Rashed, and T. Shanableh, "Agroaid: A mobile app system for visual classification of plant species and diseases using deep learning and tensorflow lite," *Informatics*, vol. 9, no. 3, p. 55, Jul. 2022.

[36] A. Sehgal and N. Kehtarnavaz, "Guidelines and benchmarks for deployment of deep learning models on smartphones as real-time apps," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 450–465, Feb. 2019.

[37] M. Z. M. Shamim, "Hardware deployable edge-ai solution for prescreening of oral tongue lesions using tinyml on embedded devices," *IEEE Embedded Systems Letters*, vol. 14, no. 4, pp. 183–186, Dec. 2022.

[38] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023.

[39] D. Sinha and M. El-Sharkawy, "Thin mobilenet: An enhanced mobilenet architecture," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics; Mobile Communication Conference (UEMCON)*. IEEE, Oct. 2019.

[40] R. R. d. Souza, M. Toebe, A. C. Mello, and K. C. Bittencourt, "Sample size and shapiro-wilk test: An analysis for soybean grain yield," *European Journal of Agronomy*, vol. 142, p. 126666, Jan. 2023.

[41] Suharjito, G. N. Elwirehardja, and J. S. Prayoga, "Oil palm fresh fruit bunch ripeness classification on mobile devices using deep learning approaches," *Comput. Electron. Agric.*, vol. 188, p. 106359, Sep. 2021.

[42] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023.

[43] The jamovi project, "jamovi (version 2.5) [computer software]," https://www.jamovi.org, 2024, retrieved from https://www.jamovi.org.

[44] L. Yang, J. Fan, Y. Liu, E. Li, J. Peng, and Z. Liang, "A review on state-of-the-art power line inspection techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9350–9365, Dec. 2020.

[45] Z. Zeng, C. Huang, W. Zhu, Z. Wen, and X. Yuan, "Flower image classification based on an improved lightweight neural network with multi-scale feature fusion and attention mechanism," *Mathematical Biosciences and Engineering*, vol. 20, no. 8, pp. 13 900–13 920, 2023.

[46] Q. Zhang, X. Che, Y. Chen, X. Ma, M. Xu, S. Dustdar, X. Liu, and S. Wang, "A comprehensive deep learning library benchmark and optimal library selection," *IEEE Trans. Mobile Comput.*, pp. 1–14, 2024.

[47] Z. Zhou, Z. Song, L. Fu, F. Gao, R. Li, and Y. Cui, "Real-time kiwifruit detection in orchard using deep learning on android™ smartphones for yield estimation," *Comput. Electron. Agric.*, vol. 179, p. 105856, Dec. 2020.