

Reduced Complexity Classifier Architecture for Edge Devices in Real-Time Wildfire Detection

Pedro M. R. Pereira, Felipe A. P. de Figueiredo, and Rausley A. A. de Souza

Abstract—This article presents a comprehensive approach to forest fire detection, starting with the construction of a deep learning model based on the MobileNet architecture and transfer learning. Post-training quantization is applied to make the model suitable for edge device implementation. It provides computational efficiency, reducing storage size by 64.75%, RAM usage by 58.38%, and inference time by 86.21%. Practical implementation is carried out, integrating the model into a low-cost Arduino Nano device in real time. The performance evaluation has included detailed metrics such as accuracy, recall, and F1-score, as well as tests with fire simulation videos. Despite limitations such as camera angle, the quantized model has demonstrated robust results, slightly outperforming the floating-point model. The proposal is relevant for forest fire detection and other areas, such as industries.

Keywords—Computer vision, deep learning, edge devices, fire detection, and post-training quantization.

I. INTRODUCTION

Forest fires, especially those in fire-prone landscapes, have become increasingly frequent and severe, causing widespread destruction to biodiversity and human life [1]. These fires release harmful gases, disrupt ecosystems, and present serious public health risks [2]. Communities near affected areas are particularly vulnerable, facing heightened risks of respiratory diseases, mental health challenges, and stress-related disorders, with firefighters, especially, at risk for elevated suicide rates [3]. Each year, forest fires affect over 370 million hectares of land globally, releasing approximately 1.8 billion tons of greenhouse gases and contributing to around 340000 premature deaths from respiratory and cardiovascular conditions [4]. In addition to devastating wildlife and local habitats, forest fires contribute to the collapse of entire ecosystems [5]. The urgency for comprehensive strategies to protect nature and human populations is clear, especially as these fires can spread to urban areas, compounding the danger. Timely and efficient fire detection is critical for mitigating these impacts.

Traditional fire detection methods have advanced to overcome the limitations of earlier, expensive techniques like spectroscopy and particle sensors. The authors of [6] developed a system combining color and motion analysis to accurately identify fire pixels in video sequences, employing a Gaussian-smoothed color histogram and temporal intensity variations to reduce false alarms. Additionally, in [7], a two-stage RGB-based method for early fire detection that leverages fire pixel growth and color dynamics is introduced. Recent studies by [8] and [9] highlighted the effectiveness of simple color-space classifiers and unsupervised methods,

demonstrating that even basic rule-based approaches can provide reliable fire detection in still images, enhancing efficiency for deployment on lightweight platforms.

Integrating advanced machine learning (ML) and Deep Learning (DL) techniques in fire detection has shown significant improvements over traditional methods. The authors of [10] pioneered the use of Support Vector Machines (SVM) in remote sensing, creating a specialized kernel that incorporates textural and radiometric characteristics to better classify forest fire-affected areas in satellite imagery, effectively addressing the challenges of non-linear separability in high-dimensional data. Subsequently, a hybrid method is introduced in [11], combining conventional image processing with convolutional neural network (CNN), utilizing an adaptive pooling technique to enhance feature learning, achieving an accuracy of 90.7% and reducing false detection rates. In recent work [12], a hybrid attention-based few-shot learning algorithm, ABPNet, is introduced for forest fire smoke detection with limited annotated data. Integrating a Convolutional Block Attention Module (CBAM) with meta-learning strategies, ABPNet enhances the model's ability to differentiate smoke from visually similar patterns, yielding superior performance metrics. Meanwhile, an efficient DL architecture, namely KutralNext, suitable for resource-constrained environments, achieving an average test accuracy of 84.36% across multiple fire datasets, is designed in [13]. [14] presented FFireNet, leveraging transfer learning with MobileNetV2, achieving an 98.42% accuracy in forest fire detection. Further, [15] introduced SmokeyNet, integrating CNNs and LSTMs for real-time smoke detection, and [16] proposed LW-FIRE, a lightweight model for UAV deployment. Very recently, [17] enhanced fire detection capabilities with their Dual Fire Attention Network (DFAN), combining attention mechanisms to reduce false alarms and improve inference speed.

Our study presents a novel DL-based approach to forest fire detection by leveraging MobileNet, an efficient CNN architecture tailored for constrained computing devices. While previous contributions have explored various ML and DL techniques, such as SVMs and hybrid models combining image processing with CNNs, our work treats fire detection as a straightforward classification task. We utilize a quantized model designed for real-time execution on resource-constrained devices. This approach allows the deployment of low-cost, scattered sensors, promotes scalability in sensor networks, and minimizes reliance on centralized infrastructure, which is particularly beneficial for remote areas. Our model's quantization reduces storage size by 64.75%, RAM usage by 58.38%, and inference time by 86.21%, enhancing computational efficiency.

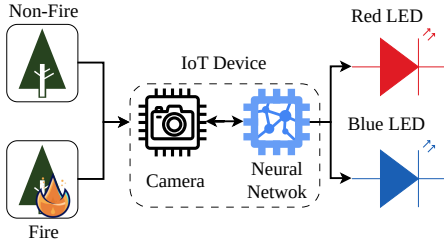


Fig. 1. Simplified diagram of the fire detection system.

 TABLE I
DIVISION OF THE DATABASE INTO TRAINING AND TESTING SETS.

	Fire	Non-fire	Total
Training	1667	1657	3324
Testing	424	418	842
			4166

II. MATERIALS AND METHODOLOGY

A. Materials

The integration of the complete system is depicted in Fig. 1. The hardware and software used in this study include the Arduino Nano 33 BLE Sense platform, featuring a 32-bit ARM Cortex-M4 CPU, and the OV7675 optical sensor, optimized for capturing VGA images with high sensitivity in low-light conditions.

B. Data Processing

An efficient forest fire detection system begins with acquiring a database representative of the problem.

The dataset containing forest fire images used in this work is a compilation of other databases and is publicly available¹. During preprocessing, duplicate images were excluded, and the training and test sets were split at a 80%:20% ratio. The final division of the sets is shown in Table I.

Next, the images undergo resizing so that the different dimensions of the images that constitute each set are standardized to a resolution of 96×96 pixels. Finally, the images are normalized by multiplying them by a scale factor of $1/255$ so that the value of all pixels is continuously distributed in the interval $(0, 1)$, ensuring greater model learning capacity.

C. MobileNet

MobileNet is a lightweight and efficient CNN architecture designed specifically for computer vision applications on mobile and embedded devices. It achieves remarkable performance while maintaining low computational complexity and memory requirements, making it suitable for devices with limited resources. Traditional CNNs employs standard convolutions, simultaneously performing a single convolution operation across all input channels. In contrast, MobileNet employs depthwise separable convolutions, which divide the standard convolution into two distinct steps: depthwise convolution and pointwise convolution [18]. However, it is important to note that the complete MobileNet model with its 28 layers and full depth is unviable due to this project's

¹<https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images>



Fig. 2. Reduced complexity classifier architecture for edge devices, supporting float-32 and quantized int-8 implementations.

 TABLE II
MODEL ARCHITECTURE.

Layer	Output Shape	# Parameters
Input	(96, 96, 3)	0
MobileNet 25%	(3, 3, 1280)	410208
Reshape	(9, 1280)	0
Dense	(9, 16)	20496
Dropout	(9, 16)	0
Flatten	(144)	0
Dense	(2)	290
Total:		430994

hardware constraints. Instead, it was chosen to employ only 25% of its depth, maintaining effectiveness in extracting essential features while accommodating resource limitations.

D. Classifier Architecture

The final model adopts an architecture composed of several layers, as illustrated in Fig. 2. After preprocessing, the input layer is configured to receive images in the format $(96, 96, 3)$, which indicates a resolution of 96×96 pixels and three RGB color channels. The input network then processes the MobileNet image, producing an output with dimensions $(3, 3, 1280)$. Subsequently, the MobileNet's output undergoes a reshape operation, preserving the outermost dimension and reshaping the internal dimensions, resulting in a shape of $(9, 1280)$. Then, a dense layer with 16 neurons, operating only on the outer dimension, is applied, yielding an output of $(9, 16)$. A dropout layer with a 10% rate is introduced to promote regularization and prevent overfitting. Next, a Flatten layer transforms the output into a one-dimensional representation of the shape (144) . This output is directed to another dense layer with two neurons, which aggregates information and allows differentiation between the two desired classes: the presence (decision 1) or the absence (decision 0) of a forest fire. Table II presents the model architecture, detailing the output dimensions of each layer and the number of parameters involved. Moreover, the architecture's ability to process input images in real time facilitates continuous monitoring, enabling timely detection and response. Furthermore, this architecture's compact size makes it highly scalable, enabling deployment across many devices, particularly those with constrained computational power. Its computational resources and memory footprint efficiency enable seamless integration into a wide range of edge devices, from inexpensive to expensive.

E. Training and Transfer Learning

The detection model training process incorporates transfer learning. This approach leverages the knowledge acquired by a pre-trained model on a more general task and applies it to a related specific task. MobileNet, previously trained on the ImageNet dataset, serves as the base model in this project.

The initial stage of training involves applying a data augmentation policy, which introduces variations in the training images, such as mirroring, brightness adjustment, and random resizing. This strategy aims to enrich the training dataset, providing the model with diversified exposure to patterns and contexts and strengthening its generalization ability.

During training, a new training set is created, reserving 20% for validation. Initially, the MobileNet layer is frozen, so its weights are not updated. The Adam optimizer is used, with a learning rate of 0.0005 and the categorical cross-entropy as the loss function. The accuracy metric is monitored. This stage is conducted over 20 epochs, with batches containing 32 images.

After this initial phase, fine-tuning is implemented to adapt the model to the specific characteristics of the forest fire detection dataset. The model is reloaded from the point of best performance during the initial training. At this stage, the percentage of trainable layers is adjusted, unfreezing 65% of the topmost layers. This process allows the model to adapt more specifically to the nuances of forest fire detection, while the previous layers remain frozen to preserve MobileNet's prior knowledge. The model is then recompiled and trained for another 10 epochs with a learning rate of 0.000045.

Once trained, the model is not yet ready for implementation because it was trained using 32-bit floating-point (float-32) operations. These operations are costly for microcontrollers, leading to high energy consumption and slow operation. Thus, post-training quantization was performed on the model. Full quantization was chosen, meaning the weights and activation functions are converted to 8-bit integers (int-8) to avoid floating-point operations on the microcontroller [19].

Full integer quantization requires calibration, which involves estimating the range of values for each floating-point tensor in the model. This calibration step ensures the quantized values accurately represent the original floating-point values. A representative dataset, a small subset of the randomly chosen training data, calibrates the model. This process uses TensorFlow Lite, an extension of the TensorFlow library optimized for devices with limited resources.

After calibration, the model is converted to a fully quantized format using only integer operations. This format significantly reduces model size and computational complexity, lowering latency and memory usage and making it suitable for implementation on edge devices. The representation of quantized tensors in floating point can be written as $r = (q - z)s$, where r is the real value, q is the quantized value, z is the zero point, and s is the scale factor. For the aforementioned model, the scale factor is $s = 0.00390625$, and the zero point is $z = -128$.

F. Implementation

After quantization, the model can be implemented in the Arduino IDE using C/C++ code compatible with Arduino development boards based on ARM processors. The application's operation is briefly described in Algorithm 1. The forest fire detection algorithm begins by initializing the OV7675 camera and LED pins to indicate fire detection and the

Algorithm 1: Fire Detection Model

Require: OV7675 Camera, quantized model

```

1 begin
2   Initialize the OV7675 camera;
3   Configure LED pins to indicate fire detection;
4   while Interrupt == False do
5     Wait for 2 seconds;
6     Capture image from OV7675 camera;
7     Resize captured image to model input
      dimensions;
8     Normalize the image;
9     Execute the fire detection model on the
      processed image;
10    Get classification results from the model;
11    if Fire class probability > Threshold then
12      Turn on the red LED to indicate fire
        detection;
13      Turn off the blue LED;
14    end
15    else
16      Turn on the blue LED;
17      Turn off the red LED;
18    end
19  end
20  Deinitialize the OV7675 camera;
21 end

```

threshold. In this work, the detection threshold is selected empirically. We have found that a more flexible threshold of 0.6 reduces the number of false positives compared to 0.5. The main loop continuously captures an image from the camera, resizes it to the model's input dimensions, normalizes it, runs the fire detection model on the resized image, and returns the classification results. If the probability of the fire class exceeds the chosen threshold, the red LED is turned on to indicate fire detection, while the blue LED is off. Otherwise, the blue LED is on, and the red LED is off. The loop continues until interrupted and ends when the user inputs a signal to stop the inference. The algorithm then deinitializes the camera and exits.

III. RESULTS

This section presents the results obtained while evaluating the proposed forest fire detection model. The necessary codes for reproducing the project, a demonstration video, are publicly available². The training and validation losses decreased over epochs during training, demonstrating the model's ability to generalize to unseen data. Similarly, the evolution of accuracy over epochs indicated the model's robust performance, with training and validation accuracy peaking at 95.5% and 95.2%, respectively, at epoch 27.

²The project's repository can be found at <https://github.com/ErosloursViv/wilde-fire-detection>. The model can also be easily reproduced using platforms such as Edge Impulse, although this is not a requirement. <https://studio.edgeimpulse.com/public/309592/latest>.



Fig. 3. Example of classifications for the quantized model. The images displayed in this figure are the preprocessed ones used as input for the model.

TABLE III
CLASSIFICATION METRICS FOR FLOATING POINT AND INTEGER MODELS.

float-32 model					
	Class	Precision	Recall	F1-score	Support
	Fire	0.953	0.958	0.955	424
	Non-fire	0.957	0.952	0.954	418
Accuracy				0.955	842
Macro Avg.		0.955	0.955	0.955	842
Weighted Avg.		0.955	0.955	0.955	842
quantized model					
	Class	Precision	Recall	F1-score	Support
	Fire	0.976	0.950	0.963	424
	Non-fire	0.951	0.976	0.963	418
Accuracy				0.963	842
Macro Avg.		0.963	0.963	0.963	842
Weighted Avg.		0.964	0.963	0.963	842

Table III compares the results of two proposed fire detection models: the float-32 model and the quantized model on the test set. Both models adopt the same principal architecture depicted in Figure 2, with the main difference being their implementation approach: one model uses floating-point precision (float-32), while the other employs quantization to reduce precision to 8 bits (int-8). In the classification report, precision is the proportion of correctly identified positive cases among all instances predicted as positive, reflecting the model's ability to avoid false positives. Conversely, recall indicates the proportion of correctly identified positive cases out of all positive cases, highlighting the model's ability to detect true positives. The F1-score is the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives. Support refers to the number of actual occurrences of each class in the dataset. The results reveal that the quantized model outperforms the float-32 model in precision for the Fire class, achieving 0.976 compared to 0.953. However, the float-32 model slightly outperforms the quantized model in recall for the same class, scoring 0.958 compared to 0.950. This difference is reflected in the F1-score, which is practically equal for both classes in both models. Both models consistently achieve high macro-average and weighted-average accuracy, indicating robust overall classification performance. The performance difference is small but significant, given that the quantized model is much smaller and faster.

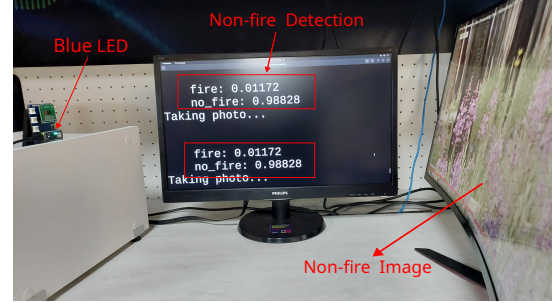


Fig. 4. Detection example for the non-fire case.

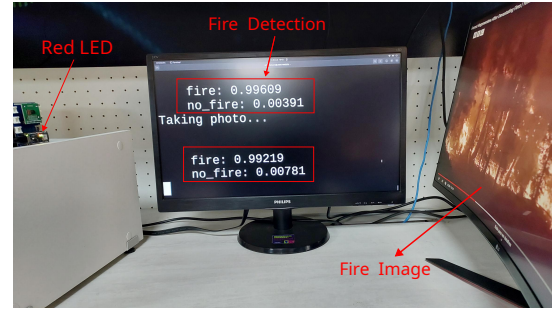


Fig. 5. Detection example for the fire case

The float-32 model requires 870.5 kbyte of storage and consumes 300.3 kbyte of RAM, resulting in an inference time of 8.909 ms. Due to its high storage and memory demands and relatively slower inference time, it is less suitable for deployment on resource-constrained edge devices. In contrast, the quantized (int-8) model requires only 307 kbyte of storage and 125 kbyte of RAM, significantly reducing the resource footprint. The quantized model achieves an inference time of just 1.229 ms. Thus, the proposed model is faster and achieves a 64.75% reduction in storage size, a 58.38% reduction in RAM usage, and an 86.21% reduction in inference time.

Fig. 3 displays the results of running the quantized classifier on 10 random images from the test set. The classifier correctly identified 9 images, of which 8 had high confidence scores. In subjective analysis, the image in the second column and second row was correctly classified, but with low confidence. The light source in the image has confused the model, resulting in a confidence of 55.86%. Similarly, the incorrectly classified image in the fourth column and first row has intense reflections at the bottom of the figure.

It could also be argued that the smoke at the top might have been mistaken for clouds. The quantized classifier is robust, generalizes to unseen data, and accurately classifies a wide range of images with high confidence. However, its performance requires improvement when handling light sources.

Replicating a real wildfire is challenging, so, for performance verification, tests were conducted using simulation videos on a 34-inch screen (0.86 meters) placed 0.60 meters from the sensor. The sensor in the experiment lacks a wide-angle lens, limiting its ability to classify images from farther away accurately. As illustrated in Fig. 4, the system correctly identified a non-fire situation, activating the blue LED. Similarly, Fig. 5 demonstrates the system's ability to detect the presence of fire, triggering the red LED.

IV. CONCLUSION

This paper presents a forest fire detection model based on DL, leveraging the MobileNet architecture with transfer learning and post-training quantization. Training and validation on a diverse dataset encompassing various environmental conditions, geographical regions, and times of day have demonstrated robust performance in distinguishing between forest fire and non-fire images. The quantized model has achieved high precision, recall, and F1-score for the fire and non-fire classes, with a marginal improvement in accuracy over the float-32 model. Practical tests using simulated fire videos validated the system's ability to classify fire events correctly.

Despite these promising results, certain limitations have been identified, including constraints on camera field of view and challenges in real-world testing environments. Future work should explore wide-angle cameras and sensor integration to enhance detection capabilities. Developing a non-empirical process for selecting detection thresholds could significantly improve the algorithm's performance.

Although the current approach uses visible image sensors, infrared sensors, and cameras are potential alternatives. Infrared technology may perform better in certain conditions, such as low visibility caused by smoke or nighttime detection. Future iterations of the system could explore incorporating these sensors to enhance robustness in challenging environments. Additionally, several factors may impact the system's performance, including proximity to the fire, the type of underlying fuel, and smoke obstruction of the fire.

Since the proposed device is cost-effective, it can be deployed extensively to cover large areas. Furthermore, integrating communication modules, such as LoRa, would enable the creation of a mesh network, allowing for widespread, real-time detection. This network could serve as an early warning system, informing more sophisticated, centralized firefighting operations and enhancing overall responsiveness and prevention efforts.

ACKNOWLEDGMENTS

This work was partially funded by CNPq (Grant Nos. 302085/2025-4 and 306199/2025-4), by Minas Gerais Research Foundation (FAPEMIG) (Grant Nos. PPE-00124-23,

APQ-04523-23, APQ-05305-23, RED-00194-23, and APQ-03162-24), by the Brasil 6G project (01245.020548/2021-07), supported by RNP and MCTI, and by the projects XGM-AFCCT-2024-2-5-1 and XGM-AFCCT-2024-9-1-1 supported by xGMobile – EMBRAPII-Inatel Competence Center on 5G and 6G Networks, with financial resources from the PPI IoT/Manufatura 4.0 from MCTI grant number 052/2023, signed with EMBRAPII, and by FINEP (n° 1060/2 contract 01.25.0883.00).

REFERENCES

- [1] C. X. Cunningham, G. J. Williamson, and D. M. J. S. Bowman, "Increasing frequency and intensity of the most extreme wildfires on earth," *Nat. Ecol. Evol.*, vol. 8, no. 8, pp. 1420–1425, Aug. 2024.
- [2] K. Gajendiran, S. Kandasamy, and M. Narayanan, "Influences of wildfire on the forest ecosystem and climate change: A comprehensive study," *Environ. Res.*, vol. 240, p. 117537, 2024.
- [3] C. A. T. Cortes, S. Thurow, A. Ong, J. J. Sharples, T. Bednars, G. Stevens, and D. D. Favero, "Analysis of wildfire visualization systems for research and training: Are they up for the challenge of the current state of wildfires?" *IEEE Trans. Vis. and Comp. Graphics*, pp. 1–20, 2023, Early access.
- [4] J. B. Koudenoukpo, "As wildfires increase, integrated strategies for forests, climate, and sustainability are ever more urgent," 31 July 2023, UN Chronicle.
- [5] B. M. Flores *et al.*, "Critical transitions in the Amazon forest system," *Nature*, vol. 626, no. 7999, pp. 555–564, feb 2024.
- [6] W. Phillips III, M. Shah, and N. da Vitoria Lobo, "Flame recognition in video," *Pattern Recognit. Lett.*, vol. 23, no. 1, pp. 319–327, 2002.
- [7] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," in *Proc. 2004 International Conference on Image Processing, 2004. ICIP '04.*, vol. 3, 2004, pp. 1707–1710 Vol. 3.
- [8] A. Akagic and E. Buza, "A study of rule-based flame pixel classifiers in still images," in *Proc. 2022 30th Telecommunications Forum (TELFOR)*, 2022, pp. 1–4.
- [9] E. Buza and A. Akagic, "Unsupervised method for wildfire flame segmentation and detection," *IEEE Access*, vol. 10, pp. 55 213–55 225, 2022.
- [10] F. Lafarge, X. Descombes, and J. Zerubia, "Textural kernel for svm classification in remote sensing: application to forest fire detection and urban area extraction," in *Proc. IEEE International Conference on Image Processing 2005*, vol. 3, 2005, pp. III–1096.
- [11] Y. Wang, L. Dang, and J. Ren, "Forest fire image recognition based on convolutional neural network," *J. Algorithms Comput. Technol.*, vol. 13, p. 1748302619887689, 2019.
- [12] T. Li, H. Zhu, C. Hu, and J. Zhang, "An attention-based prototypical network for forest fire smoke few-shot detection," *J. For. Res.*, vol. 33, no. 5, pp. 1493–1504, Oct 2022.
- [13] A. Ayala, B. J. T. Fernandes, F. Cruz, D. Macêdo, and C. Zanchettin, "Convolution optimization in fire classification," *IEEE Access*, vol. 10, pp. 23 642–23 658, 2022.
- [14] S. Khan and A. Khan, "Ffirenet: Deep learning based forest fire classification and detection in smart cities," *Symmetry*, vol. 14, no. 10, 2022.
- [15] A. Dewangan, Y. Pande, H.-W. Braun, F. Vernon, I. Perez, I. Altintas, G. W. Cottrell, and M. H. Nguyen, "Figlib & smokeynet: Dataset and deep learning model for real-time wildland fire smoke detection," *Remote Sens.*, vol. 14, no. 4, 2022.
- [16] A. Akagic and E. Buza, "Lw-fire: A lightweight wildfire image classification with a deep convolutional neural network," *Appl. Sci.*, vol. 12, no. 5, 2022.
- [17] H. Yar, T. Hussain, M. Agarwal, Z. A. Khan, S. K. Gupta, and S. W. Baik, "Optimized dual fire attention network and medium-scale fire classification benchmark," *IEEE Trans. Image Process.*, vol. 31, pp. 6331–6343, 2022.
- [18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [19] TensorFlow, "Post-training quantization," 2026. [Online]. Available: https://www.tensorflow.org/model_optimization/guide/quantization/post_training