10th International Conference on Control Engineering &Information Technology (CEIT-2025)
Proceedings Book Series –PBS- Vol 23, pp.314-321

Design of a TinyML embedded system for vehicle recognition

Zakaria MOUTAKKI[1]*, Ali HADDI[1], Mohamed BOUSLA[1]
[1] Innovating Technologies Team, National School of Applied Sciences, Abdelmalek Essaadi University, Tetouan 93000, Morocco

Abstract

In this work, we present the design of an embedded system for vehicle recognition, comprising two parts: software and hardware. The software part is based on a TinyML algorithm for vehicle recognition. The hardware part is based on the ESP32 CAM card. Data collection, a critical stage in any artificial intelligence system, was carried out using the ImageNet database and other customized images. These images underwent a set of pre-processing techniques such as dimension reduction, before starting training with Tensorflow Lite. The resulting model will be implemented in the ESP32 CAM board to build the embedded system.

1. Introduction

Vehicle recognition via TinyML [1] represents a major breakthrough for various intelligent applications, such as: automated surveillance (real-time traffic analysis), optimized traffic management, automotive in-vehicle systems. This technology also opens up prospects in key areas such as: autonomous vehicles (environmental perception), road safety (violation detection), intelligent parking (identification of available spaces).

Tiny Machine Learning (TinyML) is a sub-discipline of Machine Learning dedicated to the deployment of AI models on energy-efficient embedded devices (microcontrollers, IoT sensors). Its key features include extreme optimization with lightweight models (a few KB) running on MCUs (e.g. Arduino, ESP32), low power consumption: runs on long-term batteries (milliwatts), local processing: No dependence on the cloud, ideal for real-time and secure applications [2]. This approach makes it possible to integrate AI into constrained environments, while maintaining high operational performance.

Implementing an artificial intelligence algorithm on a microcontroller-based board presents hardware constraints in terms of available memory and required computing power [3][4]. As a result, data size has to be minimized in order to have an adequate learning model. At the same time, recognition accuracy must be optimized, as this is a critical system whose failures can pose a threat to human life.

ISSN: 2961-6611

Researchers have opted to deploy AI models on embedded systems, an approach that offers major advantages but faces challenges that the scientific community is striving to overcome. Dziri et al [5] have developed an embedded system for tracking multiple objects in real time, incorporating an innovative occlusion management method. Their architecture is based on a Raspberry Pi equipped with a RaspiCam, combining: A detection module based on background subtraction, A tracking module using an enhanced GMPHD tracker. To resolve occlusions, their solution relies on the implementation of a distance threshold between bounding boxes to identify overlaps, and a comparison of object characteristics before/after occlusion to re-identify targets. Results were: 30 FPS at QVGA resolution and 15 FPS at VGA, demonstrating effectiveness for real-time applications. Srijongkon et al [6] designed a vehicle counting system on a Soc Xilinx FPGA, optimizing detection via adaptive background subtraction, adjusting parameters such as lighting to limit the impact of shadows and improve accuracy. This work illustrates how embedded AI can meet critical needs (tracking, counting) while overcoming the challenges of hardware constraints.

Our vehicle recognition system identifies different types of vehicles (cars, trucks, bicycles) from images or camera data. This is based on the Tensorflow Lite (TFLite) trained model [7]. TFLite is an open-source framework from Google designed to deploy Machine Learning models on embedded, mobile or IoT devices, with compute, memory and energy constraints. It is particularly well suited to TinyML projects, thanks to its light weight and optimization for real-time inference. In our vehicle recognition system, TFlite presents a suitable method because it offers certain advantages, namely: Lightweight models, low latency, compatibility and energy savings.

2. Presentation of the designed system

2.1. Hardware selection

There are a number of boards suitable for TinyML image processing applications, such as Arduino Nano 33 BLE Sense, ESP32 and Raspberry Pi Pico, because they have sufficient processing power and perhaps even link a camera module. In our case, we opted to use ESP32-CAM.

The ESP32-CAM is a popular, low-cost module combining an ESP32 microcontroller with an OV2640 (2 MP) camera, ideal for TinyML projects including image, face or vehicle recognition. Features include:

ISSN: 2961-6611

- Microcontroller: ESP32 (dual-core, 240 MHz, Wi-Fi/BLE).

- Camera: OV2640 (resolutions up to 1600x1200, compressed JPEG).

- Memory:

  - Internal RAM: 520 KB (shared with Wi-Fi).
  - External PSRAM: 4 MB (required for image processing).

- Connectivity: Wi-Fi 802.11 b/g/n, Bluetooth 4.2.

- I/O: GPIO, UART, I2C, SPI.

Figure 1. ESP32-CAM module.

Figure 2. Ov2640 camera.

The ESP32-CAM is a great choice for a TinyML vehicle recognition system because it has a built-in camera module (OV2640), Wi-Fi connectivity, and a powerful dual-core processor, all in a small, low-power package.

2.2. Artificial intelligence method selection

A vehicle recognition system can be developed using a number of different approaches. For example, we can use an object detection module after pre-processing, based on motion. Another solution may be based on approaches using sliding windows for feature computation [9]. But this kind of approach could be computationally expensive, and much larger.

The choice of TinyML (Machine Learning on embedded devices) is motivated by unique advantages tailored to the constraints of IoT systems, edge devices and applications requiring local intelligence. Here are the key reasons:

a. Energy savings

- Low power consumption: TinyML models are optimized to run on microcontrollers (e.g. ESP32) consuming a few milliwatts.

ISSN: 2961-6611

- Long battery life: Ideal for stand-alone devices (e.g. traffic sensors, surveillance cameras) running for months without recharging.

b. Low cost

- Affordable hardware: Microcontrollers costing just a few dollars (ESP32-CAM) vs. expensive servers or GPU cards.
- No cloud dependency: Avoid cloud subscriptions (AWS, Azure) and data transfer costs.

c. Real Time

- Minimal latency: Local inference without depending on an Internet connection (e.g. collision detection in < 50 ms).
- Autonomous decision: immediate response (e.g. alert when a vehicle in danger is detected).

2.3. Database preparation

a. Data collection

A specialized database of vehicle categories will be the starting point for training the model. Images will be collected from the web and from images taken in real life. The collected dataset must be labeled (cars, trucks, bikes, motorcycles) before training.

A database such as ImageNet, featuring vehicle categories, can be adopted as an alternative in case the specialized dataset is not sufficient. This is because, to build a robust module, we need a set of vehicle images taken under different conditions, namely: vehicle angle, lighting conditions, presence of occlusion. In this case, data augmentation can help solve the problem. And let's not forget that the limited memory in ESP32-CAM can restrict the size of the model. A balance has to be struck between model complexity and resource constraints.

Figure 3. Images of the database used in training.

ISSN: 2961-6611

b. Image pre-processing

Data pre-processing is important. Images need to be resized to a smaller resolution (96x96 pixels) to suit the microcontroller. Then a grayscale conversion is applied to reduce the size. Finally, pixel normalization can be considered for the same purpose of reducing the memory footprint.

Thus, the pre-processing module will contain the following steps:

- Resize images to 96x96 pixels.

- Convert to grayscale to reduce data size.

- Normalize pixel values (e.g. [0, 1]).

c. Model design and optimization

Simple and customized convolutional neural networks (CNNs) are well suited to image recognition [10]. This is also true for TinyML, as the model contains only a few layers. Techniques such as quantization have been adopted to reduce model size. Quantization converts 32-bit floating-point weights into 8-bit integers, making the model smaller and faster. TensorFlow supports the training operation. This is done by transfer to generate a pre-trained model. This saves training time and improves accuracy. Conversion to TFLite format is then performed.

2.4. Testing and optimization

Prior to hardware implementation, a series of tests were carried out to determine certain coefficients related to accuracy and performance.

a. Basic metrics

- Accuracy: Percentage of correct predictions.

- Recall: Ability to detect all instances (e.g. not missing vehicles).

- Latency: Inference time (e.g. < 100 ms for real time).

- Memory consumption: RAM/ROM used (e.g. < 80% of capacity).

b. Real-life testing

- Various scenarios: Low light, rain, odd viewing angles.

ISSN: 2961-6611

- Noise robustness: Add blur or artifacts to images.

- Hardware benchmark: Measure latency on ESP32-CAM vs Raspberry Pi.

2.5. Implementation using TensorFlow Lite Micro

Once the model has been trained and converted to TF Lite, it needs to be implemented in the ESP32 CAM board. Tools like TensorFlow Lite Micro can help. The inference code will capture the camera images, pre-process them (resizing, grayscale conversion, normalization) and feed them into the model. The output will be the predicted vehicle class.

2.6. Reducing energy consumption

Energy consumption is a problem. TinyML is supposed to be energy-efficient. The model must therefore be optimized for efficient operation. For this reason, the ESP32-CAM will be in deep sleep mode until motion is detected.

3. Conclusion

In this work, we presented the design of an embedded system for vehicle recognition. The system is based on TFLite as the core software module. This choice is proven by the quality of this approach with regard to constraints that can hinder the implementation of such a system, such as memory constraints and computing power. The system will be implemented on an ESP32 CAM board, which is ideally suited to combining an ESP32 microcontroller with an OV2640 (2 MP) camera. Looking ahead, we plan to complete the system and implement an IoT platform for sending vehicle data via the Internet, based on the MQTT protocol.

References

[1] Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, *34*(4), 1595-1623.

[2] Warden, P., & Situnayake, D. (2019). *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media.

[3] Sanchez-Iborra, R., & Skarmeta, A. F. (2020). Tinyml-enabled frugal smart objects: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, *20*(3), 4-18.

[4] Immonen, R., & Hämäläinen, T. (2022). Tiny Machine Learning for Resource-Constrained Microcontrollers. *Journal of Sensors*, *2022*(1), 7437023.

[5] Dziri, A., Duranton, M., & Chapuis, R. (2016). Real-time multiple objects tracking on Raspberry-Pi-based smart embedded camera. Journal of Electronic Imaging, 25(4), 041005.

[6] K. Srijongkon, R. Duangsoithong, N. Jindapetch, M. Ikura and S. Chumpol, "SDSoC based development of vehicle counting system using adaptive background method," 2017 IEEE

ISSN: 2961-6611

Regional Symposium on Micro and Nanoelectronics (RSM), 2017, pp. 235-238, doi: 10.1109/RSM.2017.8069172

[7] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., ... & Rhodes, R. (2021). Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems*, *3*, 800-811.

[8] Moutakki, Z., Ouloul, I. M., Afdel, K., & Amghar, A. (2018). Real-time system based on feature extraction for vehicle detection and classification. *Transport and Telecommunication*, *19*(2), 93.

[9] Lee, J., Bang, J., & Yang, S. I. (2017, October). Object detection with sliding window in images including multiple similar objects. In *2017 international conference on information and communication technology convergence (ICTC)* (pp. 803-806). IEEE.

[10] Dutta, L., & Bharali, S. (2021). Tinyml meets iot: A comprehensive survey. *Internet of Things*, *16*, 100461.

ISSN: 2961-6611