# Quantization of MobileNetV2 for Resource-Constrained Microcontrollers

Luca Lazzaroni[1], Francesco Bellotti[1], Ali Dabbous[1], Alessandro Pighetti[1], and Riccardo Berta[1]

[1] Department of Electrical, Electronic and Telecommunication Engineering (DITEN), University of Genoa, Via Opera Pia 11a, 16145 Genova, Italy
luca.lazzaroni@edu.unige.it

**Abstract.** The proliferation of IoT and edge computing devices demands for efficient deployment strategies due to their limited computational capabilities and energy constraints. This paper investigates the application of deep learning models in such resource-constrained environments, focusing on the quantization of the MobileNetV2 architecture. We evaluate three primary quantization techniques: dynamic quantization, static quantization, and quantization-aware training. Using a subset of the Vehicle Make and Model Recognition dataset, specifically the Most Stolen Vehicles in the US in 2017, we compare our results with a previous study, highlighting the advancements achieved through MobileNetV2 and its quantization process. Deployment on two STM32 boards, L4 and H7 series, demonstrates the effectiveness of the quantized MobileNetV2 model in achieving efficient, low-power, and low-latency execution on low-power MCUs.

**Keywords:** Internet of Things, Microcontrollers, MobileNetV2, Quantization, STM32, Tiny Machine Learning.

## 1 Introduction

With the extensive adoption of IoT and edge computing paradigms, smart devices are becoming pervasive in everyday life [1], including sport activities trackers [2], agriculture monitoring systems [3], target classification systems [4], structural health monitoring systems [5], tactile sensing [6], and voice assistants [7]. Due to their reduced computational capabilities and low power constraints, their efficient deployment must address a range of potential choices and relevant challenges, such as data collection [2], sensor type (e.g., [8]), hardware platform and accelerator (e.g., [9, 10]), and models, including those specific to TinyML like model pruning [11], data compression [12], and quantization [13]. Quantization is a significant TinyML technique which approximates the floating-point values of a neural network (NN) model's weights and activations to a limited set of integer numbers. Quantization aims to reduce memory usage, energy consumption, bandwidth, and computational costs, albeit with a slight reduction in accuracy. The three main quantization techniques are dynamic quantization, static quantization, and quantization-aware training (QAT) [14].

The literature documents significant achievements through quantization. [15] proposes a memory-efficient library of binary NNs with a notable reduction in memory footprint and latency. [16] presents an intermittent low-power smart camera for monitoring applications which integrates an energy harvesting system. The system includes an NN accelerator to speed up computations and save energy. [17] proposes an agile Jupyter notebook to train, validate, and deploy low-resolution image classification models in an hour, using the three aforementioned quantization techniques. In the context of NN robustness against adversarial attacks, [18] reports that quantization introduces resilience against such attacks relative to the full precision counterpart.

In this TinyML-oriented context, mobile NNs have also been proposed. Such NNs are designed specifically designed for TinyML tasks. Among them we recall MobileNet [19], MCUNet [20], and SqueezeNet [21]. While these networks are specifically designed to be reduced in size, they can still benefit from quantization to enable their deployment on MCUs with very low flash memory and RAM specifications.

This paper presents a study of the three quantization techniques applied to the MobileNetV2 [22] architecture, a model specifically designed for mobile and edge device applications [23] which enhances the previous MobileNet. Our evaluation uses a subset of the Vehicle Make and Model Recognition dataset (VMMRdb) [24], focusing on the Most Stolen Vehicles (MSV) in the US in 2017 subset. The results are compared with a previous work from Kristiani et al. [25], highlighting the advancement carried out by the MobileNetV2 architecture, which allows to obtain better accuracy results, and the quantization process, which enables the deployment of the model on low-power MCUs. Two STM32 boards are considered for deployment, namely STM32L4R9I and STM32H7B3I. The obtained energy consumption and execution time are also reported, demonstrating the effectiveness of the quantized MobileNetV2 in enabling efficient, low-power, and low-latency execution of DL models on resource-constrained devices.

## 2 Dataset

For the training of our model, we employed a subset of the Vehicle Make and Model Recognition dataset (VMMRdb) [24]. The subset, namely Most Stolen Vehicles (MSV), was generated to contain only the most stolen vehicles reported in the US in 2017. Vehicles from multiple years are categorized as stolen cars based on their exterior similarity, in order to provide a larger sample size for analysis. The resulting dataset consists of 5,709 images belonging to 10 vehicle model classes, listed as follows.

1. Honda Civic (1997 – 1998)
2. Honda Accord (1996 – 1997)
3. Ford F150 (2005 – 2007)
4. Chevrolet Silverado (2003 – 2004)
5. Toyota Camry (2012 – 2014)
6. Nissan Altima (2013 – 2015)
7. Toyota Corolla (2011 – 2013)

8. Dodge Ram 1500 (1995 – 2001)
9. GMC Sierra 1500 (2007 – 2013)
10. Chevrolet Impala (2007 – 2009)

In our work, we focus on this 10-classes subset considering an image resolution of 224 × 224. Results will be compared with [25], where the same dataset has been used for training an image classification model to be exploited on a Raspberry Pi, obtaining a 91% accuracy using a VGG16 convolutional neural network. For consistency, we used the same train-validation-test split ratio, i.e., 0.7-0.2-0.1.

Similarly to [25], we applied data augmentation to increase the diversity of the training data and improve the model's robustness. The data augmentation process, performed using the TensorFlow function "ImageDataGenerator", included random rotations up to 20 degrees, width and height shifts up to 20% of the image size, shearing transformations, zooming in and out up to 20%, and horizontal flipping. These transformations help the model generalize better by simulating different viewing conditions and variations in the dataset.

## 3 MobileNetV2

MobileNetV2 is an optimized version of the original MobileNet, designed for mobile and edge device applications. It employs depthwise separable convolutions, which replace a full convolution with a factorized version that splits the convolution operation into two separate layers: a depthwise convolution and a pointwise convolution. The architecture includes residual connections and introduces inverted residual blocks, where the input and output are thin bottleneck layers, contrasting with traditional residual connections that operate on wider layers. This significantly reduces the number of parameters and computational cost, keeping the model's ability to capture intricate patterns without significant computational overhead. Also, the α parameter in MobileNetV2 controls the width of the network. Specifically, it scales the number of filters in each layer by a factor of α. The MobileNetV2 topology is defined in Table 1.

**Table 1.** MobileNetV2 architecture."t" is the expansion factor, "c" is the number of output channels, "n" is the number of repeated iterations of the same layer, and "s" is the stride. All spatial convolutions use 3 × 3 kernels.

| Input | Operator | t | c | n | s |
|---|---|---|---|---|---|
| 224 × 224 × 3 | conv2d | - | 32 | 1 | 2 |
| 112 × 112 × 32 | bottleneck | 1 | 16 | 1 | 1 |
| 112 × 112 × 16 | bottleneck | 6 | 24 | 2 | 2 |
| 56 × 56 × 24 | bottleneck | 6 | 32 | 3 | 2 |
| 28 × 28 × 32 | bottleneck | 6 | 64 | 4 | 2 |
| 14 × 14 × 64 | bottleneck | 6 | 96 | 3 | 1 |
| 14 × 14 × 96 | bottleneck | 6 | 160 | 3 | 2 |
| 7 × 7 × 160 | bottleneck | 6 | 320 | 1 | 1 |
| 7 × 7 × 320 | conv2d1 × 1 | - | 1280 | 1 | 1 |
| 7 × 7 × 1280 | avgpool7 × 7 | - | - | 1 | - |
| 1 × 1 × 1280 | conv2d1 × 1 | - | class | - | - |

The MobileNetV2 model was implemented using TensorFlow. Learning rate was set to 0.0001 with Adam optimizer, batch size 16, and number of epochs 150. Also, the α parameter was set to 0.5 to reduce the number of filters in each layer to half of their original number.

The final model, evaluated on the test set, achieved an accuracy of 92.42%, with a model size of 5,323 KB and activation size of 1,532 KB. We therefore decided to explore quantization techniques to meet the specifications of resource-constrained devices, which typically possess no more than 2 MB flash memory and 500 KB RAM [26].

# 4 Quantization

The three typical quantization techniques were employed and compared, i.e., dynamic quantization, static quantization, and QAT.

## 4.1 Dynamic Quantization

The most straightforward quantization method is dynamic quantization, where all model parameters (weights and biases) are converted to 8-bit integers. The scale factor and zero-point (i.e., the integer value to which the real value zero is mapped) values necessary for executing quantization [27] can be computed before inference, as the each parameter range is known after training. Consequently, computations are performed using efficient 8-bit-integer matrix multiplications and convolutions. In dynamic quantization, scale and zero-point values for activations are calculated on-the-fly during inference, meaning the activations are stored in 32-bit float registers and converted to and from integers before and after layer operations (multiplication or convolution).

## 4.2 Static Quantization

Static quantization involves the quantization of variable data (i.e., input, output, and activations). Scales and zero-points for all variable data can then be pre-computed by feeding batches of data through the network and computing the resulting activations' distributions. The drawback is that if the data samples are unrepresentative, the computed scales and zero-points may not reflect the true inference scenario, thus threatening accuracy.

## 4.3 Quantization Aware Training

While quantization allows to reduce the memory footprint and the power needed to perform inference, it however introduces information loss, resulting in lower accuracy compared to 32-bit floating-point models. Nevertheless, this loss can be estimated by computing the difference between the outcomes of a 32-bit floating-point NN and its quantized counterpart and incorporating this difference into the model's loss function

during training. Therefore, all weight adjustments during training can be made with the awareness that the model will ultimately be quantized, a concept known as QAT.

### 4.4 Quantization methods comparison

We applied dynamic and static quantization to the model described in Section 4. For QAT, since weight adjustments are made during training, we trained the model from scratch keeping the same architecture. Results are listed in Table 2.

**Table 2.** Comparison of the three quantization methods.

| Quantization method | Accuracy (%) | Model size (KB) | Activation size (KB) |
|---|---|---|---|
| No quantization | 92.42 | 5,323 | 1,532 |
| Dynamic | 88.43 | 1,554 | 1,532 |
| Static | 87.98 | 1,554 | 390 |
| QAT | 92.23 | 1,561 | 390 |

The baseline model without quantization achieves the highest accuracy of 92.42%, but its large sizes (5,323 KB model size and 1,532 KB activation size) make it unsuitable for resource-constrained devices. Dynamic quantization significantly reduces the model size to 1,554 KB while maintaining the activation size at 1,532 KB, which is still unsuitable for the majority of low-power MCUs. Also, this comes at the cost of reduced accuracy, which drops to 88.43%. Static quantization maintains the same reduced model size of 1,554 KB but further decreases the activation size to 390 KB. The accuracy in this case is slightly lower at 87.98%. QAT achieves a balance between accuracy and model efficiency. It results in an accuracy of 92.23%, which is very close to the baseline model, while also reducing the model size to 1,561 KB and the activation size to 390 KB. For our specific use-case, QAT reveals to be the most effective quantization method for deployment on resource-constrained devices, providing a near-baseline accuracy with significantly reduced model and activation sizes.

We compared our model with the solution proposed by Kristiani et al. in [25]. In their study, the authors observed that, among the assessed models, the most effective was a VGG16, achieving an accuracy of 91.37%, which is 0.86% lower than our QAT model. In addition, the proposed 16-bit quantized model has a size of 29.81 MB, which makes it not suitable for deployment on low-power MCUs, thus restricting the applicability of the VGG16 model to higher-end boards such as the Raspberry Pi and NVIDIA Jetson NANO.

## 5 Deployment

We deployed the QAT model on two STM boards, namely STM32L4R9I and STM32H7B3I. Results, in terms of energy consumption and execution time, are reported in Table 3.

The STM32L4R9I features 640 KB of RAM and operates at a CPU frequency of 120 MHz, while the STM32H7B3I offers a significantly larger RAM capacity of 1,400 KB and a higher CPU frequency of 480 MHz. Both MCUs have same flash memory sizes of 2,000 KB.

For the MobileNetV2 inference task, the STM32L4R9I exhibits a ~0.7× energy consumption and a ~7× execution time compared to the STM32H7B3I, reflecting its design focus on low power consumption. In contrast, the STM32H7B3I proves more suitable for tasks in which low inference times are a priority.

**Table 3.** MobileNetV2 QAT model performance on STM32 MCUs.

| Board | Energy consumption (mJ) | Execution time (ms) |
|---|---|---|
| STM32L4R9I | 57 | 2,619 |
| STM32H7B3I | 87 | 379 |

## 6 Conclusion

This study assessed the effectiveness of quantization techniques applied to the MobileNetV2 architecture for deployment on resource-constrained devices. By applying dynamic quantization, static quantization, and QAT to the MobileNetV2 model, we were able to significantly reduce the model size and activation requirements while maintaining high accuracy. The results showed that QAT achieved the best balance between model efficiency and accuracy, making it the most suitable quantization method for our use case.

The deployment on two STM32 boards, the STM32L4R9I and STM32H7B3I, highlighted the practical benefits of our approach. The energy consumption and execution time measurements confirmed that the quantized MobileNetV2 model is capable of efficient, low-power, and low-latency execution on these devices. Specifically, the STM32H7B3I board demonstrated superior performance in terms of execution time, while the STM32L4R9I board exhibited better energy efficiency, aligning with its design focus.

Our findings underscore the potential of combining advanced DL models with quantization techniques to broaden the capabilities of IoT and edge computing devices. Future work could explore further optimizations (e.g., binarization [15]) and the application of these techniques to other NN architectures and different use cases relevant to the TinyML domain.

## Acknowledgements

This work has been supported by the GEMINI "Green Machine Learning for the IoT" national research project, funded by the MUR under the PRIN 2022 program (Contract 20223M4HZ4).

## References


1. Berta, R., Bellotti, F., De Gloria, A., Lazzaroni, L.: Assessing Versatility of a Generic End-to-End Platform for IoT Ecosystem Applications. Sensors. 22, 713 (2022). https://doi.org/10.3390/s22030713.
2. Fresta, M., Bellotti, F., Capello, A., Dabbous, A., Lazzaroni, L., Ansovini, F., Berta, R.: End-to-End Dataset Collection System for Sport Activities. Electronics. 13, 1286 (2024). https://doi.org/10.3390/electronics13071286.
3. Falaschetti, L., Manoni, L., Rivera, R.C.F., Pau, D., Romanazzi, G., Silvestroni, O., Tomaselli, V., Turchetti, C.: A Low-Cost, Low-Power and Real-Time Image Detector for Grape Leaf Esca Disease Based on a Compressed CNN. IEEE J. Emerg. Sel. Top. Circuits Syst. 11, 468–481 (2021). https://doi.org/10.1109/JETCAS.2021.3098454.
4. Gianoglio, C., Rizik, A., Tavanti, E., Caviglia, D.D., Randazzo, A.: On the Edge Recurrent Neural Network Approach for Ground Moving FMCW Radar Target Classification. IEEE Trans. Consum. Electron. 70, 522–534 (2024). https://doi.org/10.1109/TCE.2023.3343460.
5. Dabbous, A., Berta, R., Fresta, M., Ballout, H., Lazzaroni, L., Bellotti, F.: Bringing Intelligence to the Edge for Structural Health Monitoring: The Case Study of the Z24 Bridge. IEEE Open J. Ind. Electron. Soc. 5, 781–794 (2024). https://doi.org/10.1109/OJIES.2024.3434341.
6. Amin, Y., Gianoglio, C., Valle, M.: Embedded real-time objects' hardness classification for robotic grippers. Future Gener. Comput. Syst. 148, 211–224 (2023). https://doi.org/10.1016/j.future.2023.06.002.
7. Lazzaroni, L., Bellotti, F., Berta, R.: An embedded end-to-end voice assistant. Eng. Appl. Artif. Intell. 136, 108998 (2024). https://doi.org/10.1016/j.engappai.2024.108998.
8. Mohankumar, P., Ajayan, J., Yasodharan, R., Devendran, P., Sambasivam, R.: A review of micromachined sensors for automotive applications. Measurement. 140, 305–322 (2019). https://doi.org/10.1016/j.measurement.2019.03.064.
9. Merenda, M., Porcaro, C., Iero, D.: Edge Machine Learning for AI-Enabled IoT Devices: A Review. Sensors. 20, 2533 (2020). https://doi.org/10.3390/s20092533.
10. Caronti, L., Akhunov, K., Nardello, M., Yıldırım, K.S., Brunelli, D.: Fine-grained Hardware Acceleration for Efficient Batteryless Intermittent Inference on the Edge. ACM Trans. Embed. Comput. Syst. 22, 82:1-82:19 (2023). https://doi.org/10.1145/3608475.
11. Wang, Z., Liu, X., Huang, L., Chen, Y., Zhang, Y., Lin, Z., Wang, R.: QSFM: Model Pruning Based on Quantified Similarity Between Feature Maps for AI on Edge. IEEE Internet Things J. 9, 24506–24515 (2022). https://doi.org/10.1109/JIOT.2022.3190873.
12. Signoretti, G., Silva, M., Andrade, P., Silva, I., Sisinni, E., Ferrari, P.: An Evolving TinyML Compression Algorithm for IoT Environments Based on Data Eccentricity. Sensors. 21, 4153 (2021). https://doi.org/10.3390/s21124153.
13. Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M., Blankevoort, T.: A White Paper on Neural Network Quantization, http://arxiv.org/abs/2106.08295, (2021). https://doi.org/10.48550/arXiv.2106.08295.
14. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper, http://arxiv.org/abs/1806.08342, (2018). https://doi.org/10.48550/arXiv.1806.08342.

15. Sakr, F., Berta, R., Doyle, J., Capello, A., Dabbous, A., Lazzaroni, L., Bellotti, F.: CBin-NN: An Inference Engine for Binarized Neural Networks. Electronics. 13, 1624 (2024). https://doi.org/10.3390/electronics13091624.
16. Nardello, M., Caronti, L., Brunelli, D.: Intermittent Intelligent Camera with LEO sensor-to-satellite Connectivity. In: Proceedings of the 11th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems. pp. 79–85. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3628353.3628550.
17. Berta, R., Dabbous, A., Lazzaroni, L., Pau, D., Bellotti, F.: Developing a TinyML Image Classifier in a Hour. IEEE Open J. Ind. Electron. Soc. 1–15 (2024). https://doi.org/10.1109/OJIES.2024.3451959.
18. Duncan, K., Komendantskaya, E., Stewart, R., Lones, M.: Relative Robustness of Quantized Neural Networks Against Adversarial Attacks. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207596.
19. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, http://arxiv.org/abs/1704.04861, (2017). https://doi.org/10.48550/arXiv.1704.04861.
20. Lin, J., Chen, W.-M., Lin, Y., cohn, john, Gan, C., Han, S.: MCUNet: Tiny Deep Learning on IoT Devices. In: Advances in Neural Information Processing Systems. pp. 11711–11722. Curran Associates, Inc. (2020).
21. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, http://arxiv.org/abs/1602.07360, (2016). https://doi.org/10.48550/arXiv.1602.07360.
22. Dong, K., Zhou, C., Ruan, Y., Li, Y.: MobileNetV2 Model for Image Classification. In: 2020 2nd International Conference on Information Technology and Computer Application (ITCA). pp. 476–480 (2020). https://doi.org/10.1109/ITCA52113.2020.00106.
23. Bertazzoni, S., Canese, L., Cardarilli, G.C., Nunzio, L.D., Fazzolari, R., Re, M., Spanò, S.: Design Space Exploration for Edge Machine Learning Featured by MathWorks FPGA DL Processor: A Survey. IEEE Access. 12, 9418–9439 (2024). https://doi.org/10.1109/ACCESS.2024.3352266.
24. Tafazzoli, F., Frigui, H., Nishiyama, K.: A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition. Presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2017).
25. Kristiani, E., Yang, C.-T., Huang, C.-Y.: iSEC: An Optimized Deep Learning Model for Image Classification on Edge Computing. IEEE Access. 8, 27267–27276 (2020). https://doi.org/10.1109/ACCESS.2020.2971566.
26. Svoboda, F., Fernandez-Marques, J., Liberis, E., Lane, N.D.: Deep learning on microcontrollers: a study on deployment costs and challenges. In: Proceedings of the 2nd European Workshop on Machine Learning and Systems. pp. 54–63. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3517207.3526978.
27. Wu, H., Judd, P., Zhang, X., Isaev, M., Micikevicius, P.: Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation, http://arxiv.org/abs/2004.09602, (2020).