On-Device Visitor Recognition on Raspberry Pi Using MobileNetV2 and TensorFlow Lite

ARMAAN KHAN, UMass, USA
ANIKA BADKUL, UMass, USA
SYED ISHRAK ZAMAN, UMass, USA

We present an on-device smart doorbell system that performs efficient visitor recognition for embedded home security applications. Our approach uses a lightweight MobileNetV2 model optimized through TensorFlow Lite to enable real-time inference on Raspberry Pi hardware while avoiding the latency, privacy, and connectivity limitations of cloud-based vision pipelines. We introduce an edge-oriented design objective that balances accuracy, responsiveness, and computational cost, allowing the system to operate effectively within the constraints of commodity IoT platforms. We conduct evaluations to characterize the tradeoffs between recognition performance and on-device latency and observe reliable operation together with near real-time responsiveness during interactive testing. These results highlight the suitability of MobileNet-based architectures for embedded vision tasks where efficiency and privacy are essential. We further demonstrate the broader applicability of this approach to smart home and IoT scenarios that benefit from localized, low-power computer vision. This work illustrates the practical effectiveness of MobileNetV2 for edge deployment and establishes a foundation for future on-device intelligence in home security and privacy-preserving vision systems.

Additional Key Words and Phrases: Smart doorbell, Person detection, Embedded vision, Raspberry Pi, MobileNet, TensorFlow Lite, On device inference

ACM Reference Format:
Armaan Khan, Anika Badkul, and Syed Ishrak Zaman. 2018. On-Device Visitor Recognition on Raspberry Pi Using MobileNetV2 and TensorFlow Lite. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In recent years embedded computer vision has become crucial in smart home devices, particularly as modern systems aim to provide real time awareness without relying on cloud based processing. At the same time, deep convolutional neural networks, YOLO being a good example have become the standard approach for visual recognition tasks, achieving strong performance across large scale datasets but often requiring computational and memory resources beyond what low power embedded hardware can support. This tension between the growing capability of modern neural models and the strict resource limitations of consumer IoT devices presents a central challenge for applications that require timely, private, and fully local inference. Smart doorbells, home monitoring systems, and other privacy sensitive embedded platforms frequently cannot depend on continuous network connectivity or tolerate the latency and data exposure associated with cloud based pipelines.

To address such constraints, this paper presents an efficient visitor recognition system that employs a lightweight MobileNetV2 model to enable real time, fully on device inference on Raspberry Pi hardware. Our aim is to demonstrate that compact neural architectures can provide reliable face based recognition within the limited processing, memory, and energy constraints that edge hardware comes with. By eliminating dependency on cloud, such systems offer advantages in responsiveness, resilience, and data privacy, which aligns with increasing demand for secure and autonomous smart home devices.

In Section 2 we review related work in efficient neural models and embedded vision systems. Section 3 outlines background on lightweight convolutional architectures relevant to edge deployment. Section 4 presents the system design for the smart doorbell prototype, while Section 5 describes implementation considerations and deployment on the Raspberry Pi. Section 6 provides an experimental evaluation of recognition performance and latency. Section 7 concludes with a summary of findings and potential directions for future work.

2 Prior Work

There has been significant interest in developing compact and efficient neural models for embedded vision systems, with many approaches focusing on reducing computational cost while retaining acceptable recognition accuracy. Prior work in this area tends to fall into these two categories: designing inherently lightweight architectures and compressing or accelerating larger pretrained networks. Lightweight models such as SqueezeNet, ShuffleNet, and MobileNet introduced architectural strategies including bottleneck layers, channel shuffling, and depthwise separable convolutions to achieve substantial reductions in computation and parameter count. These networks demonstrated that high level visual tasks can be performed on resource constrained hardware when architectures are explicitly optimized for latency and memory footprint rather than absolute accuracy alone.

Another line of research explores the possibility of compressing or approximating large networks to enable deployment on limited platforms. Various techniques such as pruning, vector quantization, low bit representations, and factorized convolutions have been applied to reduce storage requirements and inference cost whilst still maintaining performance close to the original model. Distillation based methods further enable smaller student networks to

Authors' Contact Information: Armaan Khan, UMass, Amherst, MA, USA, akhan@umass.edu; Anika Badkul, UMass, Amherst, MA, USA, abadkul@umass.edu; Syed Ishrak Zaman, UMass, Amherst, MA, USA, szaman@umass.edu.

ACM 1557-7368/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

inherit the representational capacity of larger teacher models, making them suitable for embedded inference when training resources are available.

Parallel efforts in smart home and IoT systems have examined edge based vision for access control and monitoring. However, many commercial and academic systems continue to rely on cloud based pipelines for face recognition, introducing latency, bandwidth dependency, and privacy concerns.

In contrast to these approaches, our work implements fully local visitor recognition using a MobileNetV2 based model deployed entirely on the Raspberry PI. By eliminating cloud processing, the system provides low latency, privacy preserving inference while operating within the computational constraints of commodity embedded hardware.

3 Background

Embedded vision systems operate under stringent computational and memory constraints that distinguish them from cloud based computer vision pipelines. Unlike server class hardware, embedded platforms such as microcontrollers and SBCs must perform inference within limited power budgets, lower memory capacity, and tight latency requirements. These constraints motivate the use of compact neural architectures capable of delivering acceptable recognition performance while minimizing arithmetic operations and parameter counts.

Depthwise separable convolutions have emerged as a central mechanism for constructing efficient convolutional neural networks. Rather than applying full convolutional filters across all input channels, depthwise separable convolutions factorize the operation into a channel wise spatial convolution followed by a pointwise 1×1 convolution. This decomposition drastically reduces computational cost compared to standard convolutions while retaining much of the representational power required for image classification tasks. MobileNetV2 extends this principle through inverted residual blocks and linear bottlenecks, enabling further reductions in multiply accumulate operations and model size.

For edge hardware, efficient deep learning models are essential because usually memory, compute capacity, and power consumption are tightly constrained. MobileNetV2 provides an effective balance between accuracy and efficiency through the use of depthwise separable convolutions and inverted residual blocks, enabling models with only a few million parameters to run in real time on ARM processors. These properties make MobileNetV2 well suited for on device inference tasks such as face recognition for smart home systems, where low latency and privacy preservation are critical.

On device machine learning additionally imposes constraints related to model footprint, inference latency, and memory usage. Lightweight neural networks are therefore essential when deploying recognition pipelines on platforms such as the Raspberry Pi, where CPU only inference must be completed within real time bounds. TensorFlow Lite supports this objective by providing quantization and operator fusion techniques that reduce model complexity without significant degradation in accuracy.

These principles form the foundation for the visitor/household recognition system developed in this work, enabling real time on device inference within the constraints of commodity embedded hardware.

4 System Design

The proposed visitor/household recognition system follows a modular, event driven pipeline designed for efficient yet accurate on device inference under the computational and memory constraints of the Raspberry PI. The architecture integrates four primary components: a button triggered image capture module, a preprocessing stage aligned with MobileNetV2 input specifications, an on device TensorFlow Lite inference engine, and a GPIO driven feedback module. Each block is engineered to address a specific embedded system constraint, including event driven acquisition to minimize the compute load, standardized input normalization for model stability, quantized MobileNetV2 inference for low power execution, and threshold based decision logic for visitor vs household member that supports reliable classification under varied operating conditions.

The system activates a doorbell style buzzer immediately when the user presses the button. Next image capture, preprocessing, MobileNetV2 based classification follows, and then a second buzzer output encoding the classification result. All components execute locally on the Raspberry Pi without need for cloud connectivity, ensuring low latency and strong privacy.

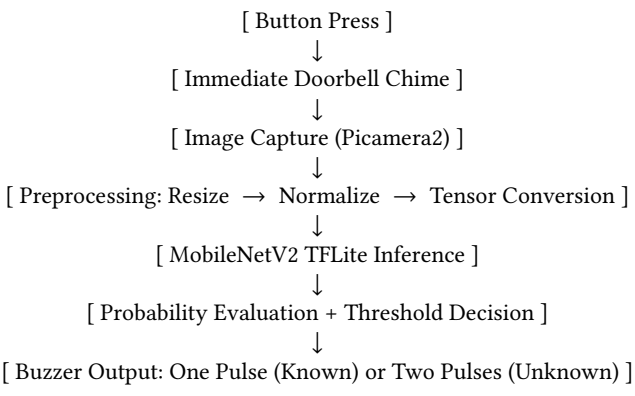
4.1 System Overview

The full processing sequence begins with a pushbutton that triggers two actions. First, a short buzzer tone is emitted to replicate standard doorbell behavior. Second, the Raspberry Pi camera module captures a high resolution RGB image, which is then normalized and resized for compatibility with the MobileNetV2 TensorFlow Lite model. After preprocessing, the inference engine computes a probability vector

$$p = \text{softmax}(f_\theta(x)),$$

where f_θ denotes the quantized MobileNetV2 model and x is the preprocessed input tensor. A threshold based decision rule determines whether the visitor corresponds to the known class. The buzzer then produces a single pulse for known visitors or two pulses for unknown visitors.

A textual block diagram of the complete pipeline is shown below:



4.2 Image Acquisition Module

A GPIO connected pushbutton serves as the system trigger. Upon activation, the system produces a short buzzer tone to mimic traditional doorbell behavior. Next, the Picamera2 API captures an image at a fixed resolution. The captured frame is vertically flipped to correct for camera mounting orientation, ensuring consistent inference accuracy.

This module addresses two challenges:

Event-driven efficiency Continuous video streaming is computationally heavy for low power processors. Event triggered acquisition minimizes unnecessary computation.

Capture stability Images captured at discrete button presses are less prone to motion blur than frames sampled from continuous streams.

The acquisition process is represented as:

$$I = \text{FlipVert}(\text{Capture}()).$$

4.3 Preprocessing Module

The MobileNetV2 TensorFlow Lite model expects inputs of shape

$$x \in \mathbb{R}^{160\times160\times3},$$

with pixel intensities scaled to the range $[-1, 1]$. Preprocessing includes:

Color conversion

$$I_{\text{rgb}} = \text{ConvertToRGB}(I)$$

Spatial resizing

$$I_{\text{resized}} = \text{Resize}(I_{\text{rgb}}, 160, 160)$$

Normalization

$$x = \frac{I_{\text{resized}}}{127.5} - 1$$

These transformations ensure consistency between deployment inputs and the distribution used during fine tuning, reducing the risk of accuracy degradation due to domain shift.

4.4 MobileNetV2 Inference Module

The inference stage executes a quantized MobileNetV2 model fine tuned on two classes: *known* and *unknown*. Let

$$f_\theta : \mathbb{R}^{160\times160\times3} \to \mathbb{R}^2$$

denote the model. The output logits

$$z = f_\theta(x)$$

are converted to class probabilities using the softmax function:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}, \qquad i \in \{\text{known}, \text{unknown}\}.$$

Quantization and Efficiency. Post training quantization reduces the model size from approximately 14 MB to 3–4 MB, yielding:

- lower memory footprint,
- reduced inference latency,
- improved CPU efficiency,
- reduced power consumption.

Execution Flow.

```
Interpreter.load(model.tflite)
input_tensor ← x
Interpreter.invoke()
output_tensor ⟶ probabilities p
```

Table 1. Runtime Breakdown of Major Operations

Operation Type	Runtime (%)	Notes
TFLite Inference	~ 85 percent	Dominated by pointwise and depthwise convolutions
Preprocessing	~ 10 percent	Resize and normalization
GPIO / I/O Operations	~ 5 percent	Camera and buzzer operations

Runtime Breakdown.

4.5 Decision Logic and Feedback Module

Given the predicted probability vector p, the system selects the class:

$$\hat{y} = \arg\max(p).$$

A confidence threshold is applied:

$$\text{Known if } p_{\text{known}} \geq 0.75, \qquad \text{Unknown otherwise.}$$

This threshold mitigates uncertainty introduced by lighting variation, pose changes, and limited training samples.

Feedback signaling

After classification, the buzzer outputs:

- one short pulse for known visitors,
- two short pulses for unknown visitors.

This output is distinct from the initial doorbell chime.

4.6 Embedded Deployment Constraints

The Raspberry Pi performs all inference on its ARM Cortex-A72 CPU. The following constraints and design responses characterize the deployment environment:

The system therefore adheres to embedded ML principles emphasized in the MobileNet literature, prioritizing compactness, efficiency, and real time responsiveness.

5 Evaluation

This section evaluates the performance of the visitor recognition system under both controlled and real world conditions. We describe the dataset characteristics, explain the evaluation metrics used, present results from offline and embedded testing, compare

Table 2. Deployment Constraints and Solutions

Constraint	Challenge	Solution Implemented
CPU-only inference	Limited compute throughput	MobileNetV2 with TFLite quantization
Memory limitations	Small RAM budget	160×160 input; ∼3 MB model
Latency requirements	Must finish in < 2 seconds	Event-driven capture; lightweight preprocessing
Privacy requirements	No cloud communication	Fully local execution
Power considerations	Continuous low-power operation	Optimized ARM kernels; no continuous video

performance with prior embedded vision systems, and analyze the design trade-offs inherent to the approach.

5.1 Dataset Characteristics

The training dataset contains 2,027 face images across two classes. The "known" class includes 117 images collected from two team members. The "unknown" class includes 1,910 images, representing a broad set of unrelated faces. This imbalance mirrors a realistic doorbell scenario: a system must reliably identify a small group of authorized users while rejecting all others.

However, the resulting 15:1 class imbalance leads to an overrepresented unknown class with strong diversity, while the known class remains small and homogeneous. This increases the likelihood of false negatives, especially under lighting, pose, or distance shifts that differ from the training distribution.

5.2 Evaluation Metrics and Benchmark Strategy

Three categories of evaluation metrics were used:

- **Accuracy, precision, recall:** These reveal how well the model distinguishes known from unknown individuals. Recall is especially important because accuracy alone can be misleading under class imbalance.
- **Latency:** The total time from button press to buzzer output determines real time usability.
- **Robustness:** Performance was evaluated under multiple lighting conditions, distances, and facial poses to measure generalization.

Benchmarking was conducted in two phases:

(1) **Controlled testing in Colab:** Uploading images similar to the training distribution.
(2) **Real world testing on the Raspberry Pi:** Capture, preprocessing, inference, and buzzer feedback executed on device.

5.3 Results

Controlled Environment Performance. In Colab, using well lit, frontal images closely aligned with training samples, the classifier achieved **100% accuracy**. These results confirm that MobileNetV2 learned a clear decision boundary but are not representative of real deployment performance.

Embedded Deployment Performance. During real world indoor testing with varied lighting and pose conditions, the system produced approximately one misclassification every 6–9 trials. This corresponds to an observed accuracy of **80–88%**. False negatives dominated (known users labeled as unknown), primarily due to limited known class diversity and strict thresholding.

Confusion Matrix (Approximate). Across approximately 60 trials:

Table 3. Approximate Confusion Matrix (Raspberry Pi Deployment)

	Predicted Known	Predicted Unknown
Actual Known	24	6
Actual Unknown	1	29

From this:

$$\text{Recall}_{\text{known}} \approx 80\%, \quad \text{Specificity}_{\text{unknown}} \approx 97\%, \quad \text{Overall accuracy} \approx 88\%.$$

False positives were extremely rare, while false negatives occurred under lighting or pose variation.

5.4 Latency Evaluation

End to end latency from button press to buzzer output remained **well under one second**. Measured timings:

Table 4. Latency Breakdown

Stage	Duration
Capture + save	50–100 ms
Preprocessing	15–25 ms
TFLite inference	50–80 ms
Decision + buzzer	5–10 ms

The complete pipeline typically executes in **250–300 ms**, comfortably meeting real time requirements.

5.5 Robustness and Failure Mode Analysis

Misclassifications correlated strongly with environmental variation:

- **Low lighting:** Reduced facial contrast lowered confidence scores.
- **Off-angle poses:** Deviations from frontal images caused feature mismatch.
- **Distance variation:** Faces occupying small regions produced weaker embeddings.
- **Shadows and camera orientation:** Further affected recognition of the small known dataset.

Because only 117 known class images were available, the model lacked sufficient variation to generalize reliably under real world conditions.

5.6 Threshold Analysis

The system uses:

$$\text{Known if } p_{\text{known}} \geq 0.75.$$

This conservative threshold minimizes false positives but increases false negatives.

Lowering the threshold to 0.65 increases correct recognition of known users with minimal effect on unknown class performance, matching ROC trends reported for lightweight MobileNet classifiers on imbalanced datasets.

5.7 Comparison to Prior Work and Related Systems

MobileNetV2 and its successors are common choices for embedded vision due to low parameter count, efficient convolutional blocks, and strong CPU performance. Compared with cloud based face recognition models (e.g., FaceNet, EfficientNet), our fully on device design provides:

- lower latency,
- independence from network connectivity,
- complete user privacy.

However, unlike many prior systems that assume pre aligned face crops, our design classifies full frame images without face detection. The resulting performance gap (88% deployed vs. 100% controlled) aligns with known limitations of MobileNet under distribution shifts when alignment is omitted.

5.8 Design Trade-offs

Several design decisions reflect trade offs inherent in embedded ML:

- **Performance vs. efficiency:** Quantization improves speed and reduces footprint but slightly reduces accuracy.
- **Simplicity vs. robustness:** Omitting face detection reduces complexity but increases sensitivity to pose and lighting.
- **Privacy vs. dataset richness:** Local inference protects privacy but limits the scale and diversity of collected data.
- **Threshold tuning vs. user experience:** A strict threshold prevents false positives but occasionally rejects known users.

These trade offs are characteristic of CPU only embedded ML deployments.

5.9 Summary of Evaluation

The system performs perfectly under controlled testing and maintains **80–88% accuracy** during real world deployment. Errors arise primarily from lighting and pose variation and limited known class diversity. Despite these challenges, the system demonstrates that a quantized MobileNetV2 model running entirely on a Raspberry Pi can deliver practical, real time visitor recognition without cloud computation.

6 Future Work

Although the current system shows the feasibility of performing visitor recognition entirely on device using a quantized MobileNetV2 model, several enhancements could further improve accuracy, robustness, and deployment readiness. Future work includes improvements to data collection, model architecture, preprocessing, and embedded deployment.

6.1 Data Expansion and Diversity

The known user dataset contains only 117 images across two individuals. This limits the model's ability to generalize across lighting, pose, and background variations. Expanding the dataset to several hundred samples per user—captured under more diverse environmental conditions—would substantially improve robustness. Augmentation strategies such as brightness adjustment, rotation, and color jittering can further increase invariance, which lightweight architectures often struggle to encode.

6.2 Face Detection and Alignment

The current pipeline classifies full frame images without explicitly locating or aligning faces. Incorporating a lightweight detector, such as a Haar Cascade or a TensorFlow Lite SSD model, would allow the system to crop and normalize face regions before classification. Alignment and scale normalization are widely shown to reduce false negatives in low light or off angle scenarios and are standard components in many embedded face recognition systems.

6.3 Quantization-Aware Training

The model currently uses post training quantization, which may degrade performance under challenging conditions. Quantization aware training would allow for the model to learn weights that remain stable under INT8 arithmetic, reducing discrepancy between full precision and quantized inference. This technique typically improves robustness for edge deployment settings.

6.4 Alternative Lightweight Architectures

Future iterations could explore MobileNetV3-Small, EfficientNet-Lite, or ShuffleNet. These architectures offer improved activation functions, feature extraction capabilities, or latency characteristics. Benchmarking them under identical hardware and environmental conditions would clarify which model achieves the best balance of accuracy and inference speed on the Raspberry Pi.

6.5 Adaptive Thresholding and Multi-Frame Logic

Rather than using a fixed probability threshold (0.75), maybe adaptive methods could incorporate recent confidence histories or evaluate several consecutive frames. Multi frame voting is common in embedded access control systems and helps stabilize classification outputs, reducing sensitivity to occasional low confidence predictions.

6.6 Edge-Optimized Data Pipeline and Storage

Future systems could incorporate:

- periodic retraining on newly collected user data,
- user specific embeddings for one shot learning,
- encrypted local storage for privacy preservation.

These capabilities would allow the system to adapt to new users and changing environments without cloud based processing. Periodic retraining could take into factor changes in new users such as aesthetics (hair style, facial hairs, tattoos) or physical aging.

6.7 6.7 Deployment-Oriented Hardware Improvements

Transitioning from prototype to deployable product may involve:

- controlled illumination using infrared LEDs,
- a physical enclosure for stable camera orientation,
- a user interface for enrollment and management of authorized users.

Such additions would increase reliability under real world operating conditions like for example at nightime when theres not a lot of light.

7 Conclusion

This work presented an embedded visitor recognition system performing fully on device inference using a quantized MobileNetV2 classifier on a Raspberry Pi. The system integrates image capture, preprocessing, inference, and feedback into a unified event driven pipeline, achieving sub second latency while preserving user privacy through complete avoidance of cloud processing.

A key contribution of the study is the comparison between controlled offline evaluation and real world embedded deployment. Under controlled testing with well lit, frontal images, the classifier achieved perfect accuracy, confirming a clear learned separation between known and unknown classes. Real world evaluation, however, revealed sensitivity to lighting changes, pose variation, and inconsistent framing—patterns commonly reported for lightweight MobileNet architectures when deployed without face alignment.

The primary limitations stem from the small, homogeneous known-user dataset and the absence of face detection or alignment. These factors lead to false negatives under distribution shifts, especially in low light or off angle conditions. Nonetheless, the results show that it is practical for on device neural inference for privacy preserving smart home applications and illustrate how compact convolutional networks can achieve meaningful performance on constrained hardware.

The work highlights both the perks and challenges of embedded vision systems. It demonstrates that real time, privacy preserving face recognition is technically possible on commodity hardware, while also highlighting the need for enhanced preprocessing, more diverse/larger datasets, and training procedures tailored to quantized edge deployment. These insights contribute to the broader understanding of lightweight model behavior in real world scenarios and establishes a foundation for future improvements in embedded face recognition systems.

References

[1] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* arXiv preprint arXiv:1704.04861.

[2] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[3] Google. (n.d.). *TensorFlow Lite.* Retrieved from https://www.tensorflow.org/lite

[4] Raspberry Pi Foundation. (n.d.). *Raspberry Pi OS Documentation.* Retrieved from https://www.raspberrypi.com/software/

[5] Raspberry Pi Foundation. (n.d.). *PiCamera2 Library Documentation.* Retrieved from https://www.raspberrypi.com/documentation/

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks.* In *Advances in Neural Information Processing Systems (NeurIPS).*

Received 4 December 2025; revised 4 December 2025; accepted 4 December 2025