Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Real-Time Joint Classification and Heat-Source Localization on Native 8×8 Thermal Frames with a Lightweight Dual-Head CNN for Microcontroller Deployment

Abbas Yasin
Independent University, Bangladesh
Abdur Rouf
abdur230rouf@gmail.com
Independent University, Bangladesh
Protik Barua
Independent University, Bangladesh
Md. Yousuf
Independent University, Bangladesh
Raqibul Hasan
Independent University, Bangladesh

Article

Keywords: Real-time inference, Thermal sensor array, Lightweight CNN, Multi-task learning, Embedded deployment

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10203811/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Real-Time Joint Classification and Heat-Source Localization on Native 8×8 Thermal Frames with a Lightweight Dual-Head CNN for Microcontroller Deployment

Abbas Yasin · Abdur Rouf (corresponding author) · Protik Barua · Md. Yousuf · Raqibul Hasan

Received: date / Accepted: date

Abstract Indoor monitoring with RGB cameras records identifiable imagery, which is hard to justify in privacy-sensitive spaces. Ultra-low-resolution infrared arrays such as the Panasonic AMG8833 Grid-EYE avoid this at the hardware level: an 8×8 grid of 64 temperature values cannot identify a person, yet still carries enough structure to separate an empty room, an occupant, and a fire event. Prior work on this sensor class has relied on handcrafted pipelines that rarely address joint recognition and localization at native resolution. We present JFilterLocalizationCNN, a compact dual-head convolutional network (∼115,365 parameters) that classifies the thermal scene into three categories and regresses the (x, y) position of the dominant heat source in one forward pass over the raw 8×8 frame. A lightweight J-filter removes the ambient offset through per-frame background subtraction; per-frame normalization stabilizes training without distorting the spatial signatures localization depends on. On 12,000 balanced frames, the model reaches 99.75% classification accuracy and 0.39-pixel mean absolute localization error. Each inference costs roughly 1.5 million multiply-accumulate operations and 2.7 ms on a desktop CPU, within the 100 ms budget of a 10 Hz sensor. Post-training INT8 quantization compresses the model from 466 KB to 126 KB with no measurable loss of accuracy or localization precision. The same INT8 TensorFlow Lite model runs on a Seeed XIAO ESP32-S3 with the AMG8833 and reaches 81.52% accuracy (macro-F1 = 0.82) and 1.39-pixel mean localization error on 4,752 live frames. Cross-dataset experiments and a twelve-case ablation further characterize generalization and the contribution of each design choice.

Keywords Real-time inference · Thermal sensor array · Lightweight CNN · Multi-task learning · Embedded deployment

Abbas Yasin · Abdur Rouf (corresponding author)
Department of Computer Science and Engineering, Independent University Bangladesh, Dhaka, Bangladesh
E-mail: abdur230rouf@gmail.com

Protik Barua · Md. Yousuf · Raqibul Hasan

1 Introduction

Demand for indoor monitoring has grown across occupancy-aware building management, assisted living, and early fire warning [1,2,3]. Camera-based indoor deployments remain widespread, yet RGB sensing captures faces and other identifying features; in hospitals, care homes, and private dwellings that trade-off is uncomfortable, and privacy-protection requirements have sharpened the tension between sensing utility and personal privacy [4].

Low-resolution infrared arrays offer a different starting point. The Panasonic AMG8833 Grid-EYE, a thermopile sensor that streams an 8×8 grid of 64 temperature readings over I^2C at up to 10 Hz [5], cannot form an identifiable image: a person appears as a warm blob a few pixels across, and a flame as a localized hot spot. The resolution is extremely coarse by any conventional standard. That limitation is also the appeal: privacy is enforced by the optics and the pixel count, not by reversible software anonymization applied after the fact [6].

The open question is whether so little spatial information can still support useful perception. Studies on AMG8833-class hardware have largely relied on handcrafted features (adaptive thresholding, bicubic interpolation, statistical distribution fitting) to read these

tiny grids [1,6]. Deep learning, which reshaped high-resolution vision, remains thinly explored at this resolution, and the few learned approaches target segmentation or occupancy rather than joint event classification and coordinate-level localization [8]. A second constraint compounds the first: the target platform is a microcontroller. For the Journal of Real-Time Image Processing, the relevant questions are therefore not only accuracy but computational cost per frame, memory footprint, achievable frame rate, and whether the model survives the quantization needed to run on a few hundred kilobytes of on-chip memory.

This paper addresses that combined problem. We design a custom dual-head convolutional neural network that operates on the raw 8×8 grid, classifies the scene (No Object, Object, Object with Fire), and predicts the location of the dominant heat source, within a parameter budget small enough for embedded inference. The single shared backbone and two thin heads keep both tasks tied to one forward pass, which is what makes the design attractive for a real-time, resource-constrained setting rather than only for offline benchmarking.

The contributions of this work are as follows:

- A J-filter preprocessing stage that subtracts the per-frame background offset and normalizes each frame to $[0, 1]$, making the model insensitive to ambient temperature drift across recording sessions while preserving the relative heat structure required for localization.
- JFilterLocalizationCNN, a purpose-built dual-head CNN (∼115,365 parameters) whose three convolutional blocks exactly exhaust the 8×8 spatial extent ($8 \to 4 \to 2 \to 1$) and feed shared features into parallel classification and localization heads in a single inference pass.
- A real-time and embedded characterization (MAC count, per-frame latency against the 10 Hz budget, FP16/INT8 quantization, TensorFlow Lite export, and live validation on an ESP32-S3–AMG8833 node) rather than accuracy alone.
- An empirical study on 12,000 balanced frames reporting 99.75% accuracy and 0.39-pixel localization error, complemented by cross-dataset generalization on two external thermal-array corpora and a twelve-configuration ablation that isolates the role of each design choice.

The remainder of the paper is organized as follows. Section 2 reviews low-resolution thermal sensing, lightweight architectures, detection and localization, multi-task learning, and embedded inference. Section 3 describes the dataset, the J-filter pipeline, the network, and the multi-task objective. Section 4 details the experimental setup. Section 5 reports classification, localization, real-time and embedded behavior, cross-dataset generalization, and an ablation study. Section 6 discusses limitations, and Section 7 concludes.

2 Related Work

We group prior work into five strands: low-resolution thermal sensing for indoor monitoring, lightweight convolutional architectures, object detection and localization, multi-task learning, and embedded inference. Each strand is mature in isolation; the gap lies in their intersection.

2.1 Low-resolution thermal sensing

Several groups have used infrared arrays for privacy-preserving indoor sensing. Newaz and Hanada recognized human activities from an infrared array through interpolation and mathematical feature extraction, avoiding deep learning entirely [6]; the handcrafted features kept the system tied to the specific activities it was tuned for. Lin and Zhao worked directly with AMG8833 data for occupancy monitoring and positioning, combining interpolation, adaptive thresholding, and blob detection with a speed-responsive sliding window [1]. Their results show that 8×8 arrays can support indoor positioning, yet the pipeline is built from traditional image processing and handles occupancy and position only, not event classification or fire.

At slightly higher resolution, Márquez et al. classified activities and detected falls among older adults from a 32×24 array using skeletonization [2], an approach that assumes more spatial detail than an 8×8 grid provides. Newaz and Hanada later detected falls from the statistical distribution of thermal signatures, using Earth Mover's Distance as the discriminator [7]; the method is effective for binary fall/non-fall decisions but does not extend to multi-class scenes or localization. Sousa et al. examined thermal infrared sensing for fire detection, but their scope is full thermal cameras in outdoor and UAV settings, a different hardware class [3]. Closest in spirit, Naser et al. applied an encoder–decoder network to thermal-array data for human segmentation and occupancy with adaptive sensor placement [8]; that work targets segmentation, not fire recognition, and uses a heavier model than an 8×8 input warrants.

A pattern runs through this literature. Studies either apply handcrafted features to low-resolution arrays or apply deep learning to higher-resolution thermal cameras. None pairs a learned CNN with native 8×8 input for joint classification and localization. Table 1 makes the gap explicit.

Table 1 Comparison of low-resolution thermal sensing approaches. Cls.: classification target; Loc.: localization output

Study	Res.	Method	Cls.	Loc.
Newaz [6]	8×8	Statistical	Activity	No
Lin [1]	8×8	Blob det.	Occup.	Pos.
Márquez [2]	32×24	Skeleton.	Falls	No
Newaz [7]	8×8	EMD stat.	Falls	No
Sousa [3]	High-res	Therm. img.	Fire	No
Naser [8]	IR array	Enc.–dec.	Segm.	No
Ours	**8×8**	**CNN**	**3-class**	**Coord.**

2.2 Lightweight convolutional architectures

Compact network design has been driven by mobile and edge deployment. Simonyan and Zisserman showed that stacking small 3×3 kernels reaches the receptive field of larger ones with more non-linearity and fewer parameters [9], a principle our backbone inherits. He et al. enabled very deep training through residual connections [10], though such depth is unnecessary on an 8×8 input whose spatial extent is gone after three pooling steps. A line of mobile architectures followed: MobileNetV2 introduced inverted residuals with linear bottlenecks [11], MobileNetV3 refined them through architecture search [12], ShuffleNet and ShuffleNet V2 used group convolution and channel shuffle [13,14], GhostNet produced feature maps with cheap linear operations [15], and EfficientNet proposed compound scaling of width, depth, and resolution [16]. Pelee demonstrated real-time detection on mobile hardware with conventional convolutions [17]. Notably, Ma et al. argued that FLOPs alone poorly predict latency, and that hardware-aware design matters more [14].

These designs share an assumption: the input is a standard-resolution RGB image, typically 224×224 or larger. On an 8×8 single-channel grid the spatial dimensions are so small that depthwise separable factorization yields little saving while adding complexity, and compound scaling has no resolution headroom to trade. Adapting any of them to this geometry would require redesign rather than scaling, which motivates a custom network.

2.3 Object detection and localization

Modern detection follows two paradigms. Two-stage detectors such as Faster R-CNN use a region proposal network before classification and box refinement [18]; single-stage detectors such as YOLO and SSD predict labels and coordinates directly in one pass [19,20]. Feature Pyramid Networks added top-down pathways for multi-scale detection [21]. Wu et al. examined the interaction of classification and localization and found that the two prefer different head types: fully connected layers favor classification, convolutional layers favor localization [22], which directly supports a separated dual-head design.

These frameworks assume natural images, multiple objects, and box annotations. Our task differs in three ways. The input is a single-channel 8×8 grid, not a high-resolution RGB image. We regress a single (x, y) coordinate rather than bounding boxes, since a box is ill-defined for an object spanning two or three pixels. And the spatial hierarchy across $8 \to 4 \to 2$ is far shallower than feature pyramids assume. The transferable insight is conceptual: classification and coordinate regression can be solved jointly in one forward pass, and they benefit from specialized heads.

2.4 Multi-task learning

Multi-task learning shares representations across related tasks to improve generalization and efficiency. Zhang and Yang surveyed how shared features reduce overfitting when tasks are related and data is limited [23], both true here. Liu et al. proposed task-specific attention for end-to-end multi-task learning [24], and Dai et al. handled related vision tasks through a cascaded shared backbone [25]. In our setting, deciding whether a frame contains a person and locating that person both depend on the same warm region in the grid, so a shared backbone with two thin heads is a natural fit. We keep the loss simple, an unweighted sum of cross-entropy and mean squared error, rather than the uncertainty weighting or gradient normalization used when task losses sit on very different scales.

2.5 Embedded and real-time inference

Bringing inference to microcontrollers is the domain of TinyML. Banbury et al. described TensorFlow Lite Micro as a portable runtime for devices with kilobytes of RAM [26]; the runtime, however, only helps if the model is small enough to fit. That places the burden on architecture design and on post-training quantization [27]. A model for an ESP32-class device, with around 520 KB of SRAM, cannot be a shrunken desktop network; it must be designed within the budget from the outset. Closest to our deployment target, Xie et al. ran CNNs on microcontrollers with low-resolution infrared sensors for social-distance monitoring [28], underscoring that 8×8 thermal inference on commodity microcontrollers is realistic. Our work adds joint localization, an explicit

quantization and latency budget, and a generalization study across sensors.

3 Materials and Methods

3.1 System overview

The pipeline runs from raw acquisition to dual prediction. At each step the AMG8833 emits a 64-value temperature vector over I^2C. The vector passes through the J-filter stage (background subtraction, normalization, reshape) to form a single-channel $1 \times 8 \times 8$ tensor, which JFilterLocalizationCNN maps to a 3-class label and a normalized (x, y) coordinate in one forward pass. Fig. 1 places the system in its intended deployment—a ceiling- or wall-mounted node observing an indoor scene—and Figure 2 shows the end-to-end flow.

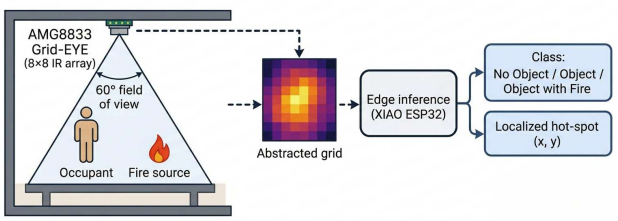


Fig. 1 Application scenario. A ceiling-mounted AMG8833 node observes an indoor space and reports one of three thermal events together with the location of the dominant heat source, without capturing identifiable imagery

3.2 Dataset

Frames were recorded indoors under three scenarios: an empty field of view (No Object), a single occupant at varying positions (Object), and a controlled heat source placed near a person (Object with Fire). Each raw frame is 64 floating-point temperatures. Ground-truth coordinates of the dominant heat source were read from the corresponding image filenames, which encode the row and column of that pixel; for the No Object class the coordinate marks the warmest background pixel. Row and column indices in $[0, 7]$ are normalized to $[0, 1]$ by dividing by 7, matching the sigmoid range of the localization head.

The corpus contains 12,000 frames balanced across the three classes (4,000 each), as summarized in Table 2. Balancing removes the class-prior confound that complicates accuracy interpretation, so the headline figure reflects genuine discrimination rather than a skewed prior. Figure 3 contrasts raw and J-filtered frames: in the raw rows the ambient offset dominates and classes are hard to separate by eye, whereas after J-filtering the No Object frames read near-uniform, the Object frames show a warm region, and the Object with Fire frames show a pronounced hot spot beside the occupant's signature.

Table 2 Class distribution of the dataset

ID	Class	Samples	Proportion
0	No Object	4,000	33.3%
1	Object	4,000	33.3%
2	Object with Fire	4,000	33.3%
	Total	**12,000**	100%

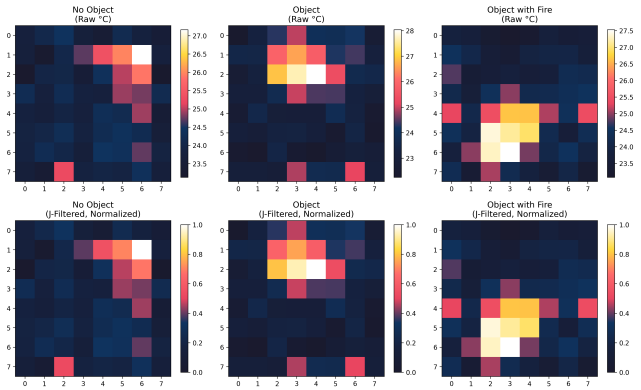


Fig. 3 Representative frames per class. Top: raw sensor output, dominated by ambient temperature. Bottom: after J-filter background subtraction, which exposes the relative thermal pattern used for both tasks. Columns: No Object, Object, Object with Fire

3.3 J-filter preprocessing

The AMG8833 reports absolute temperatures, so the same scene recorded at two ambient temperatures produces different raw values even when the relative heat pattern is identical. A model trained on absolute readings would degrade as room temperature drifts. The J-filter removes this offset by subtracting the per-frame minimum from every pixel,

$$T'(i) = T(i) - T_{\min}, \qquad T_{\min} = \min_{1 \le j \le 64} T(j), \tag{1}$$

so the coldest pixel reads zero and the rest report their elevation above that baseline. Each frame is then scaled to a common range by its own maximum,

$$T_{\text{norm}}(i) = \frac{T'(i)}{\max(T')}, \quad \max(T') > 0, \tag{2}$$

with an all-zero output retained for a perfectly uniform frame. Because Eq. (2) is a linear rescaling, it preserves

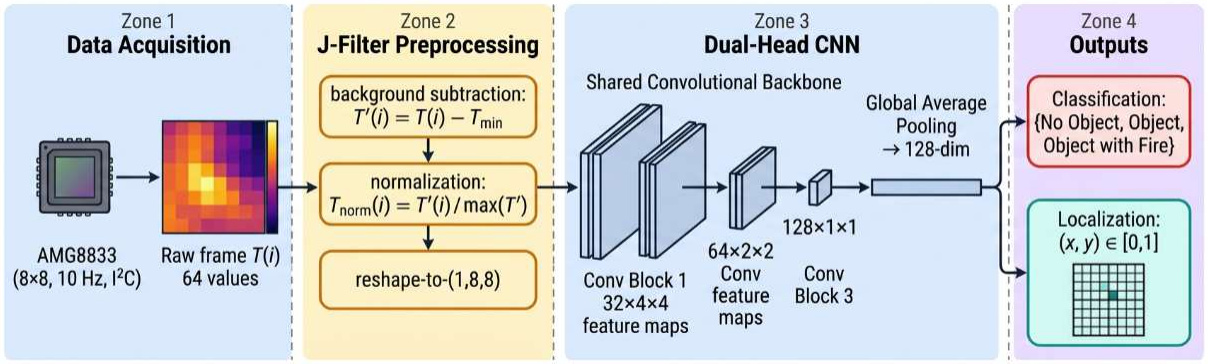


Fig. 2 End-to-end methodology. The 8×8 thermal frame is background-subtracted and normalized by the J-filter, reshaped to a $1 \times 8 \times 8$ tensor, and processed by the shared backbone; two heads emit the class label and the (x, y) coordinate in a single inference pass

the ordering and relative spacing of pixel values, leaving the spatial signature that localization relies on intact. The 64-value vector is finally reshaped into a single-channel image,

$$\mathbf{x} \in \mathbb{R}^{64} \;\rightarrow\; \mathbf{X} \in \mathbb{R}^{1\times 8\times 8}. \tag{3}$$

No temporal smoothing, Butterworth filtering, or spatial interpolation is applied; the network sees each native-resolution frame independently. This deliberately simple design keeps the preprocessing cost negligible on an embedded target, where a temporal model would add both memory and latency.

3.4 Network architecture

JFilterLocalizationCNN is a custom VGG-style dual-head network built for the 8×8 geometry. It is trained from scratch, without transfer learning, and operates at native resolution with no upscaling. A shared backbone of three convolutional blocks halves the spatial size at each stage while widening the channels ($1 \rightarrow 32 \rightarrow 64 \rightarrow 128$); a global average pooling layer then yields a 128-dimensional embedding feeding two heads. Figure 4 shows the architecture and Table 3 gives the layer-by-layer breakdown.

The first two blocks each stack two 3×3 convolutions (padding 1) with batch normalization and ReLU, followed by 2×2 max pooling. Two stacked 3×3 kernels reach a 5×5 receptive field with more non-linearity and fewer parameters than a single 5×5 kernel, valuable when every transformation must extract signal from a 64-pixel input. The third block uses a 2×2 kernel without padding to collapse the remaining 2×2 map to 1×1. Channel width rises at each stage to offset the loss of spatial resolution. Three blocks are not an arbitrary depth: they exactly exhaust the input, since $8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ leaves nothing further to pool.

Table 3 Layer-by-layer summary of JFilterLocalizationCNN

Layer	Output	Params
Input	$1 \times 8 \times 8$	–
Conv block 1		
Conv(1→32, 3×3)+BN+ReLU	$32 \times 8 \times 8$	384
Conv(32→32, 3×3)+BN+ReLU	$32 \times 8 \times 8$	9,312
MaxPool(2)	$32 \times 4 \times 4$	–
Conv block 2		
Conv(32→64, 3×3)+BN+ReLU	$64 \times 4 \times 4$	18,624
Conv(64→64, 3×3)+BN+ReLU	$64 \times 4 \times 4$	37,056
MaxPool(2)	$64 \times 2 \times 2$	–
Conv block 3		
Conv(64→128, 2×2)+BN+ReLU	$128 \times 1 \times 1$	33,152
GlobalAvgPool	128	–
Classification head		
Drop(0.3), FC(128→64), ReLU	64	8,256
Drop(0.2), FC(64→3)	3	195
Localization head		
Drop(0.3), FC(128→64), ReLU	64	8,256
Drop(0.2), FC(64→2), Sigmoid	2	130
Total trainable		**∼115,365**

Both heads take the same 128-dimensional embedding. The classification head applies dropout, a $128 \rightarrow 64$ projection with ReLU, a second dropout, and a $64 \rightarrow 3$ layer, producing logits that softmax converts to class probabilities,

$$\hat{y}_{\text{cls}} = W_2\,\text{ReLU}(W_1\,\text{Drop}_{0.3}(\mathbf{z})) \in \mathbb{R}^3. \tag{4}$$

The localization head mirrors this structure but ends in a sigmoid, which bounds the output to $[0, 1]^2$ and matches the normalized coordinate targets,

$$\hat{y}_{\text{loc}} = \sigma(W_4\,\text{ReLU}(W_3\,\text{Drop}_{0.3}(\mathbf{z}))) \in [0, 1]^2. \tag{5}$$

Bounding the coordinate by construction means the network never has to learn values outside the valid range, which smooths the localization objective. Keeping the two heads equal in depth prevents either task from

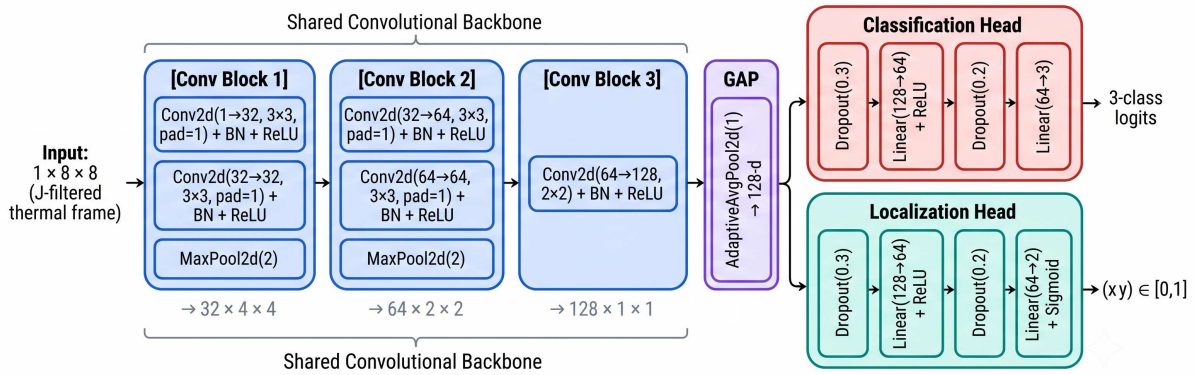


Fig. 4 JFilterLocalizationCNN. Three convolutional blocks reduce the spatial size $8 \to 4 \to 2 \to 1$ and widen channels $1 \to 32 \to 64 \to 128$; global average pooling produces a 128-dimensional embedding shared by the classification and localization heads

dominating the parameter budget; the choice is pragmatic and could be revisited if one task proved markedly harder on a future dataset.

3.5 Multi-task objective

The model is trained against a composite loss. Classification uses cross-entropy,

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{c=1}^{3} y_c \log(\hat{p}_c), \tag{6}$$

and localization uses mean squared error on the normalized coordinates,

$$\mathcal{L}_{\mathrm{MSE}} = \tfrac{1}{2} \sum_{d \in \{x,y\}} (\hat{y}_d - y_d)^2. \tag{7}$$

The total objective is their unweighted sum,

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{MSE}}. \tag{8}$$

Equal weighting is used because the two terms sit on comparable numerical scales in practice and both decay during training. We examine the alternative of an explicitly weighted loss in the ablation of Section 5.5; the simple sum avoids an extra hyperparameter without measurable cost to either task.

4 Experimental Setup

The model was implemented in PyTorch and trained on a desktop workstation. NumPy and PyTorch seeds were fixed to 42 for reproducibility. The dataset was partitioned by stratified sampling into 70% training, 10% validation, and 20% test (8,400 / 1,200 / 2,400 frames), in two sequential splits with a fixed seed; the test set is touched only for final evaluation. Optimization used AdamW [29] (learning rate 10^{-3}, weight decay 10^{-2}), which decouples weight decay from the adaptive update and gives more consistent regularization across parameter groups with different gradient magnitudes. A ReduceLROnPlateau scheduler halved the learning rate after five epochs without improvement in validation loss. Training ran for 50 epochs at batch size 32, and the checkpoint with the lowest total validation loss, not the highest accuracy, was retained, since accuracy saturates early while localization keeps improving. Table 4 lists the configuration.

Table 4 Training configuration

Setting	Value
Framework	PyTorch
Optimizer	AdamW
Learning rate	1×10^{-3}
Weight decay	1×10^{-2}
Scheduler	ReduceLROnPlateau (patience 5, factor 0.5)
Batch size	32
Epochs	50
Random seed	42
Checkpoint criterion	Lowest total validation loss

Classification quality is reported with accuracy, per-class precision, recall, and F1. Localization quality is reported as mean squared error on normalized coordinates and as mean absolute error in pixels,

$$\mathrm{MAE}_{\mathrm{px}} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(7\hat{x}_i - 7x_i)^2 + (7\hat{y}_i - 7y_i)^2}, \tag{9}$$

where the factor 7 returns normalized values to the $[0, 7]$ pixel range of the grid. For the real-time analysis we additionally report the per-frame multiply-accumulate (MAC) count and measured FP32 latency on a desktop CPU (Sect. 5.3). Post-training quantization and

Table 5 Per-class classification report on the test set

Class	Prec.	Recall	F1	Support
No Object	0.998	0.999	0.998	800
Object	0.999	0.995	0.997	800
Object with Fire	0.996	0.999	0.998	800
Accuracy	**99.75% (2,400 frames)**			

TensorFlow Lite export are evaluated on test accuracy, localization error, and stored model size (Table 6).

5 Results

5.1 Training behavior and classification

Training converges quickly and without overfitting. Classification accuracy climbs past 99% within the first few epochs and the validation curve tracks training closely throughout, while the localization error keeps decreasing across the full schedule (Fig. 5). The split in dynamics is expected: cross-entropy approaches zero once the three classes are separated, after which the total loss is governed by the localization term, which is why we checkpoint on loss rather than accuracy.

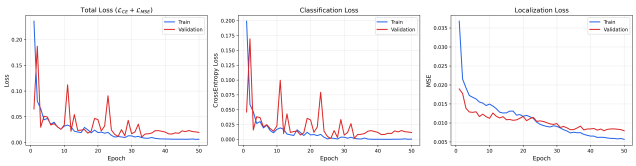


Fig. 5 Training and validation curves. Total loss and its cross-entropy and mean-squared-error components against epoch; classification saturates early while localization continues to improve

On the 2,400-frame test set the model reaches 99.75% accuracy. Per-class precision, recall, and F1 (Table 5) all exceed 0.99, and the confusion matrix (Fig. 6) shows that the few errors involve neighboring thermal patterns rather than systematic confusion between, for example, an empty room and a fire. The Object with Fire class is recognized essentially without error, which matters for the safety-relevant use case.

5.2 Localization

Coordinate regression is accurate at the scale of the grid. The model attains a normalized MSE of 0.0083 and a mean absolute error of 0.39 pixels on the 8×8 lattice, roughly a third of one pixel. Because each pixel subtends a fixed solid angle, this places the dominant heat source

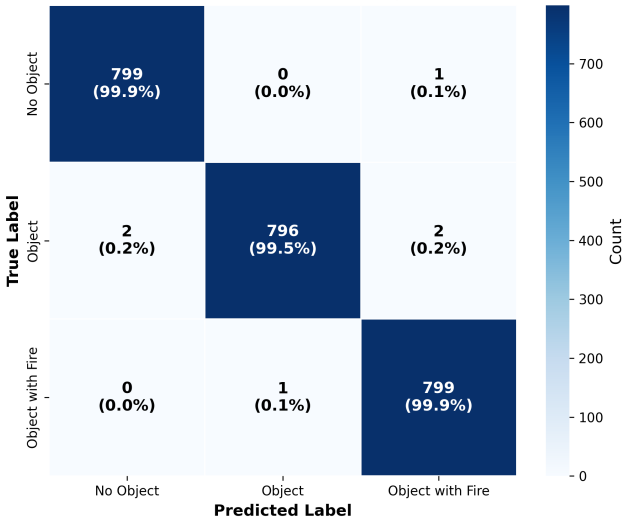


Fig. 6 Confusion matrix on the 2,400-frame test set. Off-diagonal mass is small and confined to adjacent thermal patterns

within its correct cell with margin to spare, which is the level of spatial fidelity a coarse array can meaningfully provide. Figure 7 overlays predicted and ground-truth coordinates per class; the predictions cluster tightly around the true positions across the grid. Errors are not concentrated in any single class, though the Object and Object with Fire classes carry slightly larger error than the near-static No Object case, consistent with a moving occupant whose centroid is intrinsically less stable than a fixed warm background pixel.

5.3 Real-time and embedded behavior

For a real-time target the cost per frame matters as much as accuracy. JFilterLocalizationCNN requires approximately 1.5 million MAC operations per inference, dominated by the two early 3×3 blocks. On the desktop CPU a full-precision forward pass takes 2.7 ms, which is comfortably inside the 100 ms budget implied by the sensor's 10 Hz frame rate; the J-filter adds only two passes over 64 values and is negligible by comparison. Figure 8 sketches the on-device loop: acquire a frame over I^2C, apply the J-filter, run one inference, and emit a label and coordinate before the next frame arrives.

The decisive constraint for a microcontroller is memory rather than arithmetic. We therefore evaluated post-training quantization in three formats and a TensorFlow Lite export on the test set (Table 6). FP16 halves the stored weights, and dynamic INT8 reduces them further; on the test set neither changes accuracy or localization error to the reported precision. The TensorFlow Lite INT8 model is 126 KB, a 3.7× reduction from the 466 KB

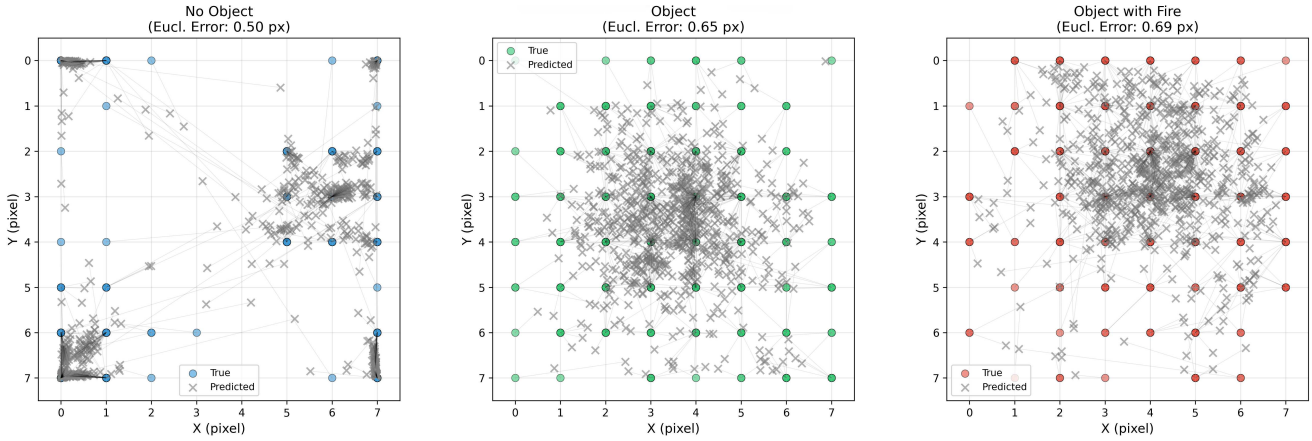


Fig. 7 Predicted versus ground-truth coordinates on the 8×8 grid, per class. Crosses mark predictions and filled circles the true positions; the close overlap reflects the overall 0.39-pixel mean absolute error (normalized MSE 0.0083)

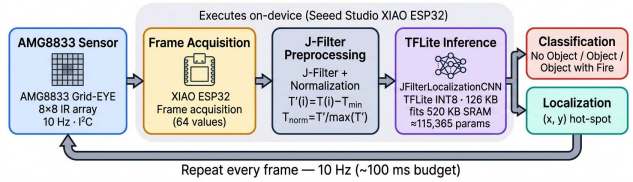


Fig. 8 On-device real-time inference loop. Acquisition, J-filter preprocessing, a single quantized inference, and output all complete within the 100 ms inter-frame interval of a 10 Hz sensor

full-precision model, and it preserves 99.75% accuracy and 0.39-pixel error. At 126 KB the network fits within the 520 KB SRAM and 4 MB flash of a Seeed XIAO ESP32-S3 device with headroom for the runtime and tensor arena. The localization head is as robust to quantization as the classifier, which is not guaranteed for small networks with tight value ranges and is a useful empirical finding for this sensor class.

The INT8 TFLite model was deployed on a Seeed XIAO ESP32-S3 paired with the AMG8833 for end-to-end validation. Two logging modes ran in parallel: raw 8×8 frames streamed to a host for archival, and on-device J-filter plus TFLite inference with predicted class label and (x, y) coordinate written to CSV. A total of 4,752 frames were collected (1,561 No Object; 1,524 Object; 1,667 Object with Fire). Table 7 and Fig. 9 summarize classification; per-class localization errors are 1.71 px (No Object), 1.74 px (Object), and 0.72 px (Object with Fire), for a mean of 1.39 px (Fig. 7 gives the controlled baseline). Overall accuracy is 81.52% with macro-F1 of 0.82. The Object with Fire class shows the smallest real-world degradation: localization error rises by only 0.03 px relative to the controlled test set (∼0.69 px → 0.72 px), because the high thermal contrast of a heat source anchors the dominant-pixel centroid even under unconstrained ambient conditions. The gap between the in-domain ceiling (99.75% accuracy, 0.39 px MAE) and these live figures reflects mounting-angle variation, ambient drift across sessions, and occupant behaviour absent from the training distribution; the live numbers therefore mark a practical deployment floor distinct from the held-out test-set ceiling.

Table 6 Quantization and deployment comparison. Test accuracy, macro-F1, localization MAE, and stored model size. The TensorFlow Lite INT8 row targets microcontroller deployment

Format	Acc. (%)	F1 (%)	MAE (px)	Size (KB)
PyTorch FP32	99.75	99.75	0.39	466
PyTorch FP16	99.75	99.75	0.39	240
PyTorch INT8	99.75	99.75	0.39	420
TFLite FP32	99.75	99.75	0.39	454
TFLite INT8	**99.75**	**99.75**	**0.39**	**126**

Table 7 Per-class classification report on 4,752 real-world frames captured during live ESP32-S3 deployment (INT8 TFLite model). Compare with the controlled test-set report in Table 5.

Class	Prec.	Recall	F1	Support
No Object	0.81	0.87	0.84	1,561
Object	0.78	0.83	0.80	1,524
Object with Fire	0.86	0.76	0.81	1,667
Accuracy	**81.52% (4,752 frames)**			
Macro avg	0.82	0.82	0.82	4,752
Weighted avg	0.82	0.82	0.81	4,752

5.4 Cross-dataset generalization

A model that only works on its own recording conditions is of limited use, so we tested transfer to two external thermal-array corpora: LINAIGE [30], an 8×8 dataset, and a 32×24 MLX90640 dataset [31]. Three

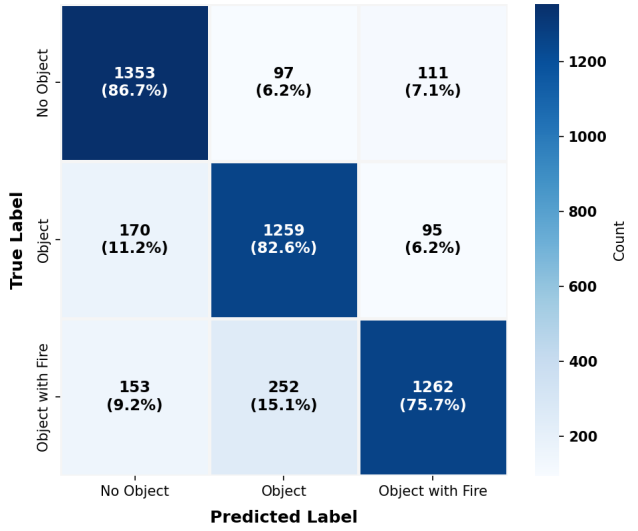


Fig. 9 Confusion matrix on 4,752 real-world frames collected during live ESP32-S3 deployment. Off-diagonal mass is higher than in the controlled test set (Fig. 6) and is concentrated in the No Object ↔ Object direction.

Table 8 Cross-dataset generalization. Accuracy under zero-shot transfer, fine-tuning, and five-fold cross-validation; the in-domain result is shown for reference

Dataset	Protocol	Acc.	Macro-F1
In-domain (ours)	3-class	99.75	99.75
LINAIGE	Zero-shot	77.00	76.91
	Fine-tuned	96.40	96.40
	5-fold CV	95.72 ± 0.39	95.72 ± 0.39
MLX90640	Zero-shot	46.09	35.60
	Fine-tuned	81.82	81.82
	5-fold CV	76.82 ± 2.53	76.76 ± 2.55

protocols were used: zero-shot inference with the frozen model, fine-tuning on the target data, and five-fold cross-validation, with results in Table 8. The fine-tuning dynamics for both corpora are shown in Fig. 10.

The pattern is informative. On the same 8×8 geometry (LINAIGE) the frozen model already classifies 77% of frames, and a light fine-tune lifts it to 96.4%, with five-fold cross-validation confirming stability at $95.7\% \pm 0.39\%$. Transfer to the 32×24 MLX90640 sensor is weaker zero-shot (46.1%), which is unsurprising given the different resolution, optics, and scene statistics; fine-tuning recovers performance to 81.8%. We read this as evidence that the learned features capture genuine thermal structure that transfers within the sensor class, while a resolution and domain shift still requires adaptation, a realistic rather than inflated account of generalization.

Table 9 Ablation study. Each row modifies one component of the full model. Acc.: test accuracy; MAE: localization error in pixels

Category	Variant	Acc. (%)	MAE
–	**Full model**	**99.75**	**0.39**
Preproc.	No J-filter	99.13	0.54
	No normalization	99.88	0.38
Arch.	No batch norm	99.33	0.45
	No dropout	99.67	0.34
	Single conv/block	99.58	0.43
	Half channels	99.63	0.51
	No conv3 + GAP	99.67	0.34
Multi-task	Classification only	99.79	1.72
	Localization only	34.88	0.14
	Weighted loss	99.79	0.26
Training	No weight decay	99.58	0.48
	No scheduler	99.71	0.37

5.5 Ablation study

To attribute performance to specific choices we removed or altered one component at a time and retrained, grouping the twelve variants into preprocessing, architecture, multi-task, and training categories (Table 9).

Several effects stand out. Removing the J-filter is the single most damaging preprocessing change, raising localization error from 0.39 to 0.54 pixels and lowering accuracy, which confirms that background subtraction, not normalization, carries the preprocessing benefit. Among architectural cuts, dropping batch normalization and halving the channels both hurt, while the half-channel model is interesting for deployment: it loses little accuracy but degrades localization, a trade a designer could accept if memory were even tighter. The multi-task rows are the most telling. A classification-only model matches our accuracy but its localization collapses to 1.72 pixels, and a localization-only model localizes slightly better yet cannot classify at all (34.9%, near chance). Joint training therefore buys accurate localization at no cost to classification, the central justification for the dual-head design. An explicitly weighted loss improves localization further (0.26 pixels) at equal accuracy, which we note as a worthwhile refinement rather than adopt by default, since it reintroduces a hyperparameter.

6 Discussion

The results support a narrow but useful claim: joint classification and localization on native 8×8 thermal frames is workable, fast, and small enough for embedded deployment, provided preprocessing preserves relative heat structure. The real-time picture is favorable on

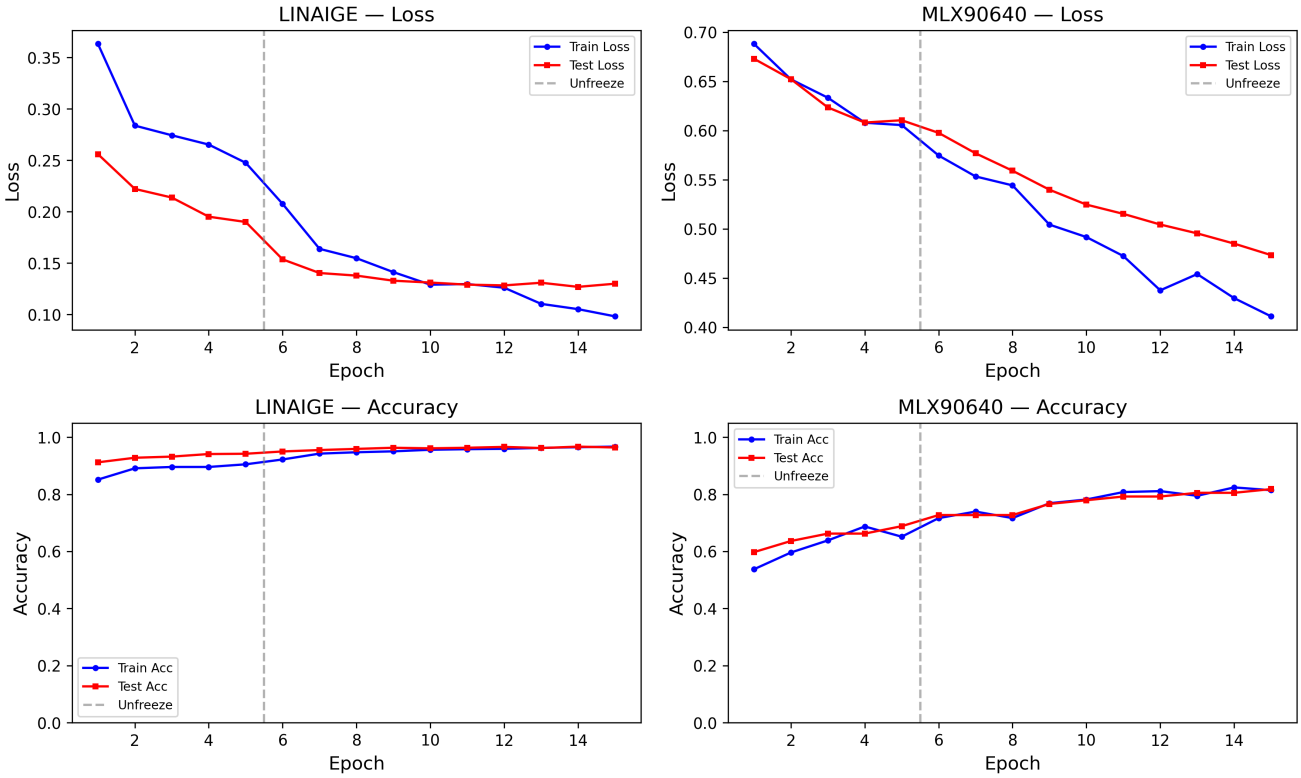


Fig. 10 Fine-tuning dynamics on the external corpora. Training and test loss (top) and accuracy (bottom) over fine-tuning epochs for LINAIGE and MLX90640; the dashed line marks the epoch at which the backbone is unfrozen. The small train–test gap indicates adaptation without overfitting

every axis the journal cares about: about 1.5 MMACs and 2.7 ms per frame against a 100 ms budget, and a 126 KB INT8 model that fits commodity microcontroller memory without accuracy loss.

Some limitations bound these conclusions. The primary dataset was collected in a single controlled indoor setting, and the fire class uses a controlled heat source rather than open flame, so the absolute numbers should be read as an in-domain ceiling; the cross-dataset study is our attempt to probe beyond that ceiling, and it shows both encouraging same-geometry transfer and the expected drop across a resolution shift. Live deployment on the ESP32-S3 provides an additional data point between the in-domain ceiling and the cross-dataset transfer results. The 18-percentage-point accuracy gap (99.75 % to 81.52 %) and the roughly three-fold localization error increase (0.39 px to 1.39 px) are consistent with the controlled-versus-real-world gap documented in embedded sensing literature: unconstrained mounting geometry, ambient temperature drift across sessions, and diverse real-world occupant behaviour all introduce noise absent from the training distribution. The Object with Fire class proves the most robust under live conditions: its localization error degrades by only 0.03 px (0.69 px to 0.72 px), demonstrating that high thermal contrast between the heat source and background is sufficient to preserve centroid accuracy even when other real-world conditions shift. The model localizes a single dominant source per frame, so multi-occupant scenes are out of present scope. Each frame is treated independently, which discards the temporal cues available at 10 Hz; a sequence model could capture entering, leaving, or falling, at some memory and latency cost on the target device. Functional on-board inference is validated on the ESP32-S3 (Sect. 5.3); per-frame cycle time on the microcontroller was not profiled separately here, so dedicated on-device latency measurement remains a natural next step alongside temporal modelling. The ablation also points to a concrete improvement (an explicitly weighted multi-task loss sharpened localization) that we report transparently rather than fold silently into the headline model.

7 Conclusion

We presented JFilterLocalizationCNN, a lightweight dual-head convolutional network that classifies an indoor thermal scene into three categories and localizes the dominant heat source directly on the raw 8×8 output of

an AMG8833 Grid-EYE sensor. A J-filter removes the ambient offset and per-frame normalization stabilizes training while preserving the spatial signatures localization needs. On 12,000 balanced frames the model attains 99.75% accuracy and 0.39-pixel localization error in a single forward pass costing about 1.5 MMACs and 2.7 ms, and post-training INT8 quantization compresses it to 126 KB, a 3.7× reduction with no measurable degradation, so it fits a microcontroller-class memory budget. Live deployment on a physical XIAO ESP32-S3 across 4,752 real-world frames yields 81.52% overall classification accuracy (macro-F1 = 0.82) and a mean localization error of 1.39 px, providing a practical performance baseline that complements the controlled-dataset ceiling. Cross-dataset experiments show that the learned features transfer within the same sensor geometry and adapt with fine-tuning across a resolution shift, and an ablation isolates background subtraction and joint training as the components that matter most. Useful perception, in other words, is achievable at a resolution most vision pipelines would discard, which is precisely what makes privacy-preserving, real-time indoor monitoring on inexpensive hardware plausible.

Declarations

Funding The authors declare that no funding was received for conducting this study.

Conflict of interest The authors declare no competing interests.

Informed consent Informed consent was obtained from the volunteer occupant involved in data collection.

Data availability The primary thermal dataset, trained model weights, training and evaluation scripts, quantization pipeline, and ESP32-S3 deployment firmware are publicly available at https://github.com/abbasyasin1n2/amg8833-jfilter-localization-cnn. The external evaluation corpora are publicly available at https://www.kaggle.com/datasets/francescodaghero/linaige (LINAIGE) and https://www.kaggle.com/datasets/rashiii3/mlx90640-thermal-image-dataset-human-and-non-human (MLX90640).

Author contributions Abbas Yasin developed the J-filter preprocessing pipeline and the JFilterLocalizationCNN model, implemented the training framework, and conducted the ablation studies and cross-dataset evaluation; he also contributed to the methodology and to the model evaluation reported in the results. Abdur Rouf developed the data-collection software and carried out the real-time implementation and embedded-system analysis, including latency measurement, post-training quantization, and live validation on the ESP32-S3–AMG8833 platform. He contributed to the system design, the dataset-collection protocol, the real-time deployment framework, and the methodology and results sections on embedded performance. Protik Barua contributed to primary data collection, performed the TensorFlow Lite conversion and deployment preparation, and drafted parts of the literature review and related manuscript sections. Md. Yousuf collected and organized the real-world deployment frames, assisted with deployment data preparation and validation, and contributed to the discussion and conclusion. Raqibul Hasan supervised the research, provided methodological guidance, and critically reviewed the manuscript and experimental design. All authors contributed to interpreting the results, reviewed the manuscript, and approved the final version for publication.

Use of AI-generated images Figs. 1, 2, 4, and 8 were generated using Google Gemini for schematic illustration; all quantitative results and experimental photographs remain original.

References

1. Lin, Y., Zhao, Q.: Human occupancy monitoring and positioning with speed-responsive adaptive sliding window using an infrared thermal array sensor. Sensors **25**(1), 129 (2025). https://doi.org/10.3390/s25010129
2. Márquez, G., Veloz, A., Minonzio, J.-G., Reyes, C., Calvo, E., Taramasco, C.: Using low-resolution non-invasive infrared sensors to classify activities and falls in older adults. Sensors **22**(6), 2321 (2022). https://doi.org/10.3390/s22062321
3. Sousa, M.J., Moutinho, A., Almeida, M.: Thermal infrared sensing for near real-time data-driven fire detection and monitoring systems. Sensors **20**(23), 6803 (2020). https://doi.org/10.3390/s20236803
4. Barsocchi, P., Calabrò, A., Crivello, A., Daoudagh, S., Furfari, F., Girolami, M., Marchetti, E.: COVID-19 & privacy: enhancing of indoor localization architectures towards effective social distancing. Array **9**, 100051 (2021). https://doi.org/10.1016/j.array.2020.100051
5. Panasonic Industry: Grid-EYE infrared array sensor (AMG8833), high performance type: reference specification. Panasonic Corporation (2019). https://industry.panasonic.com/ac/e/control/sensor/thermosensors/grid-eye
6. Newaz, N.T., Hanada, E.: A low-resolution infrared array for unobtrusive human activity recognition that preserves privacy. Sensors **24**(3), 926 (2024). https://doi.org/10.3390/s24030926
7. Newaz, N.T., Hanada, E.: An approach to fall detection using statistical distributions of thermal signatures obtained by a stand-alone low-resolution IR array sensor device. Sensors **25**(2), 504 (2025). https://doi.org/10.3390/s25020504
8. Naser, A., Lotfi, A., Zhong, J.: Adaptive thermal sensor array placement for human segmentation and occupancy

estimation. IEEE Sens. J. **21**(2), 1993–2002 (2021). https://doi.org/10.1109/JSEN.2020.3020401
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR) (2015). https://arxiv.org/abs/1409.1556
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
11. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018). https://doi.org/10.1109/CVPR.2018.00474
12. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., et al.: Searching for MobileNetV3. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324 (2019). https://doi.org/10.1109/ICCV.2019.00140
13. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6848–6856 (2018). https://doi.org/10.1109/CVPR.2018.00716
14. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: European Conference on Computer Vision (ECCV), pp. 116–131 (2018). https://doi.org/10.1007/978-3-030-01264-9_8
15. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C.J., Xu, C.: GhostNet: more features from cheap operations. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1577–1586 (2020). https://doi.org/10.1109/CVPR42600.2020.00165
16. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML), pp. 6105–6114 (2019). https://arxiv.org/abs/1905.11946
17. Wang, R.J., Li, X., Ling, C.X.: Pelee: a real-time object detection system on mobile devices. In: Advances in Neural Information Processing Systems (NeurIPS) (2018). https://proceedings.neurips.cc/paper_files/paper/2018/file/9908279ebbf1f9b250ba689db6a0222b-Paper.pdf
18. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot multibox detector. In: European Conference on Computer Vision (ECCV), pp. 21–37 (2016). https://doi.org/10.1007/978-3-319-46448-0_2
21. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944 (2017). https://doi.org/10.1109/CVPR.2017.106
22. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Rethinking classification and localization for object detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10186–10195 (2020). https://doi.org/10.1109/CVPR42600.2020.01020
23. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Trans. Knowl. Data Eng. **34**(12), 5586–5609 (2022). https://doi.org/10.1109/TKDE.2021.3070203
24. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1871–1880 (2019). https://doi.org/10.1109/CVPR.2019.00197
25. Dai, J., He, K., Li, Y., Ren, S., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3150–3158 (2016). https://doi.org/10.1109/CVPR.2016.343
26. Banbury, C., Reddi, V.J., Torelli, P., et al.: TensorFlow Lite Micro: embedded machine learning on TinyML systems. In: Proceedings of Machine Learning and Systems (MLSys), vol. 3, pp. 800–811 (2021). https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf
27. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2704–2713 (2018). https://doi.org/10.1109/CVPR.2018.00286
28. Xie, C., Daghero, F., Chen, Y., Castellano, M., Gandolfi, L., Calimera, A., Macii, E., Poncino, M., Jahier Pagliari, D.: Privacy-preserving social distance monitoring on microcontrollers with low-resolution infrared sensors and CNNs. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3205–3209 (2022). https://doi.org/10.1109/ISCAS48785.2022.9937837
29. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019). https://arxiv.org/abs/1711.05101
30. Daghero, F., Xie, C., Pagliari, D.J.: LINAIGE: low-resolution infrared array dataset for indoor monitoring. https://www.kaggle.com/datasets/francescodaghero/linaige (2022). Accessed 3 March 2026
31. Gupta, R.: MLX90640 thermal image dataset: human and non-human. https://www.kaggle.com/datasets/rashiii3/mlx90640-thermal-image-dataset-human-and-non-human (2025). Accessed 3 March 2026