Face Mask Detection: A Comparative Analysis of Deep Learning Models for Edge Deployment

Ali Asghar, Suleman Shah, Arbab Masood Ahmad
Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Peshawar, Pakistan.
*** Correspondence:** arbabmasood@uetpeshawar.edu.pk

Citation | Asghar. A, Shah. S, Ahmad. A. M "Face Mask Detection: A Comparative Analysis of Deep Learning Models for Edge Deployment.", IJIST, Vol. 8 Issue. 6 pp 2607-2628, September 2026
DOI: https://doi.org/10.33411/IJIST/2029
Received | Aug 20, 2026 **Revised |** Sep 04, 2026 **Accepted |** Sep 09, 2026 **Published |** Sep 10, 2026.

Introduction/Importance of Study: Automated face mask monitoring remains a practical requirement for respiratory-disease control in transport hubs, clinics and institutional settings, yet the deep networks that perform it accurately are rarely small enough for the microcontrollers that would host them.
Novelty statement: This work treats quantization robustness as a first-class architectural property rather than a post-hoc implementation detail, reporting FP32 and INT8 accuracy for every model under one matched protocol — with two model-specific deviations documented explicitly in the Methods — and carrying the comparison through to physical microcontroller deployment.
Material and Method: Four architectures — VGG-16, ResNet-18, MobileNetV2 and YOLOv8n-cls — were trained on a merged 19,157-image corpus of cropped face regions under a common preprocessing, optimizer and epoch budget, with two model-specific deviations documented in the Methods, three of them at a unified 96×96 edge-compatible input. Each was exported to ONNX, quantized from FP32 to INT8 with the ESP-PPQ toolchain using layer-wise equalization and learned step-size quantization (LSQ), and deployed to an ESP32-S3-CAM through ESP-IDF and ESP-DL with Haar-cascade face localization.
Results and Discussion: In FP32 all four models achieved comparable performence, reaching 95.43–97.78% accuracy with macro-F1 above 0.95 and per-class recall above 95%. Quantization reduced memory uniformly by 73–75% but degraded accuracy very unevenly: VGG-16 lost 0.67 percentage points, ResNet-18 10.44 and MobileNetV2 14.30, while YOLOv8n-cls fell by 43.20 points to 54.58%, recovering only 9 of 1,728 masked faces and falling below its majority-class baseline. Expressed as error inflation the separation is wider still, at 1.17×, 4.18×, 4.13× and 20.46× respectively. A calibration ablation showed equalization and LSQ recover a real but partial share of the loss without changing the ordering, which places the vulnerability in the four architectures tested rather than in the calibration procedure. Only MobileNetV2 and YOLOv8n-cls fitted the 8 MB PSRAM budget, at 576.03 ms and 242.20 ms per face.
Concluding Remarks: Quantization robustness, not floating-point accuracy, governs edge viability; across the four architectures tested, FP32 accuracy proved an unreliable proxy for deployed accuracy.

Keywords: Face Mask Detection; Model Quantization; Edge Computing; Convolutional Neural Networks; Microcontroller Deployment.

Introduction:

Respiratory pathogens transmitted by aerosol continue to make correct face-mask use a first-line, low-cost public health intervention. Where compliance must be monitored at scale — hospital entrances, mass-transit gates, factory floors, classrooms — manual observation is labor-intensive, inconsistent between observers, and impossible to sustain around the clock. Automated visual monitoring built on deep learning has therefore moved from a pandemic-era novelty to a standing component of environmental health surveillance, and recent reviews document a large and still-growing body of work on the problem [1].

The accuracy of these systems is, in isolation, close to solved: convolutional and transformer-based classifiers routinely exceed 95% on public mask corpora [2][3][4], and single-shot detectors report comparable numbers at video frame rates [5]. What is not solved is delivering that accuracy on the hardware where the monitoring has to happen. Practical deployments favour self-contained camera nodes over workstations or continuous cloud uplinks, both because of cost at scale and because streaming facial imagery off-device creates privacy exposure that on-device inference avoids. That pushes inference onto microcontroller-class hardware with single-digit megabytes of memory and no floating-point accelerator — the regime addressed by the tiny machine learning (TinyML) literature [6][7][8][9][10].

Bridging the two requires quantization: the conversion of weights and activations from 32-bit floating point (FP32) to 8-bit integers (INT8), which cuts memory roughly fourfold and allows inference on integer-only arithmetic units [11][12][13][14][15]. Quantization is normally presented as an approximately free compression step, with reported losses on ImageNet-scale classifiers of a fraction of a percentage point. That framing is what this study interrogates.

Tolerance for INT8 conversion depends on the dynamic range of a model's activation distributions, and the architectural features that improve floating-point efficiency — depth wise separable convolutions, activation functions, and dense split-and-concatenate feature reuse — are precisely those that can widen that range [16]. The consequence, under-reported in the applied face-mask literature, is that the ranking of architectures by FP32 accuracy need not survive quantization at all.

Existing comparative studies do not settle the question. They typically evaluate at different input resolutions, on different splits, under different augmentation regimes, and report either FP32 or post-quantization accuracy but seldom both under a matched protocol; deployment, where reported, is usually to a mobile SoC or single-board computer rather than a microcontroller, and the few studies that do compress for low-end hardware treat a single model family rather than comparing architectures [17]. Differences observed under such conditions confound architecture with experimental setup, leaving the practical question — which network should an engineer choose for a memory-bounded node? — unanswerable from the published record.

Objectives:

This study was designed to answer three questions under a single controlled protocol:

1)How do four architectures spanning the complexity spectrum — VGG-16, ResNet-18, MobileNetV2 and YOLOv8n-cls — compare in FP32 face-mask classification accuracy when trained on one corpus, at a common edge-compatible input resolution and under a closely matched optimization protocol?

2)What accuracy and memory changes follow post-training INT8 quantization of each model, and how much of any accuracy loss can be recovered by layer-wise equalization and learned step-size quantization (LSQ) rather than plain min/max calibration?

3)Which architecture, if any, satisfies the joint memory and latency constraints of an ESP32-S3-CAM while retaining usable classification accuracy?

Novelty Statement: The novelty of this work lies in treating quantization robustness as a first-class architectural property rather than a post-hoc implementation detail. Three

contributions distinguish it from the existing face-mask detection literature. First, all four architectures are compared under a closely matched protocol — one merged corpus, one train/test partition, a unified 96×96 edge-compatible input, and a common normalization, optimizer, schedule and epoch budget across the three PyTorch models — so that the reported differences are attributable primarily to architecture rather than to experimental drift. Two deviations from strict identity, MobileNetV2's additional augmentation and YOLOv8n-cls's native Ultralytics training loop, are stated in full in the Methods and their consequences for the comparison assessed there. Second, both FP32 and INT8 performance are reported for every model, with an explicit calibration ablation separating plain min/max quantization from equalization plus LSQ, quantifying how much of the penalty is recoverable without retraining the weights. Third, the comparison is carried through to physical deployment on an ESP32-S3-CAM under ESP-IDF and ESP-DL, reporting on-device latency and the memory ceiling that excludes two of the four models — a constraint desktop-simulated evaluations systematically fail to expose.

The central finding, that the most accurate FP32 model (YOLOv8n-cls, 97.78%) becomes the least usable after quantization (54.58%) while least VGG-16 proves the most robust, inverts the selection heuristic implied by accuracy-only benchmarks.

Materials and Methods:

Investigation site:

This is a computational study with no field site. All training and floating-point benchmarking were carried out at the Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Pakistan; embedded deployment and on-device timing used an ESP32-S3-CAM board in the same laboratory. The datasets are public image corpora containing no records collected from human participants by the authors.

The study proceeds in three phases, shown in the methodology flow diagram of Figure 1: fine-tuning each architecture on cropped face regions, post-training INT8 quantization against a calibration subset of the training partition, and deployment of the quantized graph to the ESP32-S3-CAM, where a Haar cascade localizes faces and the network classifies each detected region.

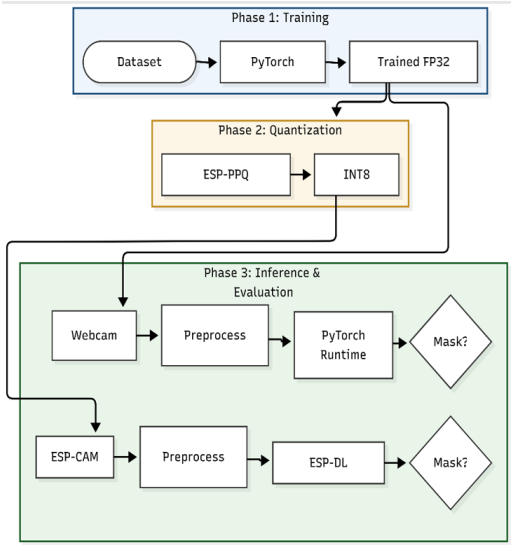

Figure 1. Methodological flow of the study: three-phase pipeline covering model training, post-training INT8 quantization, and deployment to the ESP32-S3-CAM. Phase 1 fine-tunes each architecture on cropped face regions; Phase 2 applies post-training INT8 quantization with scales calibrated on a subset of the training partition; Phase 3 loads the quantized graph onto the ESP32-S3-CAM, where a Haar cascade localizes faces and the network classifies each detected region.

Dataset Composition and Preprocessing:

A merged corpus of 19,157 face images was assembled from the MAFA masked-face collection [18] combined with WIDER FACE [19], and an auxiliary mask dataset. Both sources provide Pascal-VOC bounding-box annotations for detection; because the task here is per-face classification, every annotated box was cropped and stored under its own label. This yields a two-class corpus (with_mask, without_mask) in which each sample is a single face — the form the on-device classifier receives after Haar-cascade localization.

The two constituent sets are skewed in opposite directions: Dataset 1 (15,093 images) toward unmasked faces, Dataset 2 (4,064 images) toward masked faces. Merging therefore yields a corpus that is only moderately imbalanced overall, 7,443 masked (38.9%) against 11,714 unmasked (61.1%), and the residual skew is handled during training rather than by resampling. An 80/20 split within each source before merging gives 15,346 training and 3,811 test images, detailed in Table 1 and Figure 2.

The composition of Table 1 reconciles exactly along all three axes, and subtotal rows have been added to the table so that each reconciliation can be read from it directly. By source, Dataset 1 contributes 12,076 training and 3,017 test images, a subtotal of 15,093, and Dataset 2 contributes 3,270 training and 794 test images, a subtotal of 4,064; 15,093 + 4,064 = 19,157. By partition,12,076 + 3,270 = 15,346 training images and 3,017 + 794 = 3,811 test images; 15,346 + 3,811 =19,157, and 15,346 is 80.1% of the total. By class, 2,943 + 1,039 + 2,772 + 689 = 7,443 maskedimages and 9,133 + 1,978 + 498 + 105 = 11,714 unmasked images; 7,443 + 11,714 = 19,157, giving the 38.9%/61.1% split quoted above. Every row and column of **Table 1.** sums to its stated total, and all three axes agree on the 19,157-image corpus size.

Table Dataset composition and train/test split. Dataset 1 is the MAFA + WIDER FACE collection; Dataset 2 is the auxiliary source. Subtotal rows give each source's contribution to the merged corpus; the Combined block gives the merged totals actually used for training and evaluation. All rows and columns sum exactly, and the by-source, by-partition and by-class readings all reconcile to 19,157 images.

Subset	Split	With Mask	Without Mask	Total
Dataset 1	Train	2,943	9,133	12,076
	Test	1,039	1,978	3,017
	Subtotal	3,982	11,111	15,093
Dataset 2	Train	2,772	498	3,270
	Test	689	105	794
	Subtotal	3,461	603	4,064
Combined	Train	5,715	9,631	15,346
	Test	1,728	2,083	3,811
	Total	7,443	11,714	19,157

Reconciliation of the total: 15,093 + 4,064 = 19,157 by source; 15,346 + 3,811 = 19,157 by partition; 7,443 + 11,714= 19,157 by class.

Preprocessing was identical across the three PyTorch models except for input size: RGB conversion, resizing, scaling to [0,1], normalization with ImageNet statistics (mean [0.485, 0.456, 0.406], standard deviation [0.229, 0.224, 0.225]), and transposition from HWC to CHW. YOLOv8n-cls is trained and validated through the Ultralytics pipeline, which scales inputs to [0,1] without the ImageNet mean and standard-deviation step; that is the convention under which its pretrained weights were learned. VGG-16 alone retains a 224×224 input, because its five-stage pooling pyramid would reduce a 96×96 image to a 3×3 feature map. The identical transform is reproduced in the ONNX Runtime and ESP-DL inference paths, since any divergence — a channel-order swap in particular — produces an accuracy collapse easily

mistaken for a quantization failure. Training augmentation for VGG-16 and ResNet-18 comprised random rotation up to 20° and horizontal flipping.

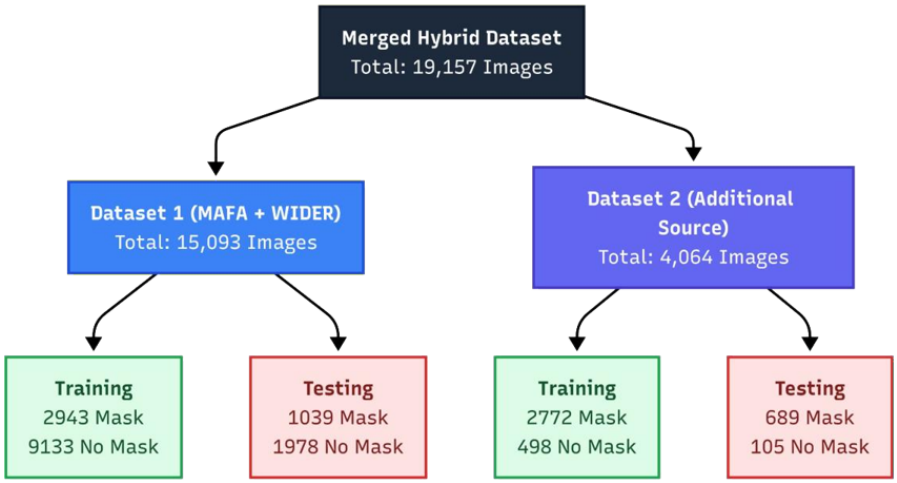

Figure 2. Composition of the merged corpus and its train/test partition across the two constituent datasets. Bars give masked and unmasked image counts for the training and test partitions of Dataset 1 (MAFA + WIDER FACE) and Dataset 2 (auxiliary source), showing the opposing class skew of the two sources and the moderate 38.9%/61.1% imbalance that survives merging. Counts correspond exactly to Table 1.

MobileNetV2 received that same base augmentation together with affine translation up to 20% of image extent, shear up to 15°, and brightness and contrast jitter of ±0.15; YOLOv8n-cls was augmented by the default Ultralytics policy. Both deviations are itemized in Table 2 and justified in the following subsection. No augmentation was applied at test time. The full hardware and software configuration for training, floating-point benchmarking and embedded deployment is given in Table 2.

Table 2. Experimental configuration for training, floating-point benchmarking and embedded deployment. Rows that differ between models are stated per model.

Component	**Specification**
Workstation	ASUS TUF Gaming F15 (FX507VI); Intel Core i7-13620H; NVIDIA RTX 4070Laptop (8 GB)
Training framework	PyTorch 2.9.1 with CUDA 12.8
Optimizer / loss	Adam, initial LR 1×10^{-4}, cross-entropy (VGG-16, ResNet-18, MobileNetV2); Ultralytics *auto* optimizer selection, initial LR1×10^{-2}, cross-entropy (YOLOv8n-cls)
LR scheduler	ReduceLROnPlateau (patience 2, factor 0.1)
Batch size / epochs	32 / 15 (best-validation checkpoint retained)
Augmentation, VGG-16 and ResNet-18	Random rotation ±20°; horizontal flip
Augmentation, MobileNetV2	As above, plus affine translation ±20% of extent, shear ±15°, brightness and contrast jitter ±0.15
Augmentation, YOLOv8n-cls	Ultralytics 8.4.6 defaults: RandAugment, mosaic, random erasing (p = 0.4), HSV jitter (0.015 / 0.7 / 0.4), scale ±50%, translation ±10%, horizontal flip
Input normalization	[0,1] scaling with ImageNet means and standard deviation (VGG-16, ResNet-18, MobileNetV2); [0,1] scaling only (YOLOv8n-cls, Ultralytics convention)
Imbalance handling	WeightedRandomSampler (PyTorch models); minority

	oversampling (YOLOv8n-cls)
Random seed	42, torch and numpy (VGG-16, ResNet-18, MobileNetV2); 0, Ultralytics default (YOLOv8n-cls). One run per model
ONNX export opset	18 (VGG-16, ResNet-18, MobileNetV2); 12 (YOLOv8n-cls)
FP32 benchmarking	ONNX Runtime 1.23.2, CPU execution provider (16 threads); OpenCV 4.13 preprocessing
Deployment target	ESP32-S3-CAM (512 KB SRAM + 8 MB PSRAM); ESP-IDF v5.1 with ESP-DL
Quantization	ESP-PPQ, per-tensor symmetric INT8; layerwise equalization + LSQ (30 epochs, LR 1×10^{-5})
Face localization	Haar cascade (OpenCV, ported to ESP32)

Deviations from a Strictly Identical Protocol:

Two aspects of the protocol are not identical across all four models. Both are stated here in full so that every comparison in the Results can be read with them in view. The first is augmentation. MobileNetV2 received affine translation, shear and brightness/contrast jitter in addition to the rotation and horizontal flipping applied to VGG-16 and ResNet-18.

This was a regularization decision, not an attempt to advantage the model. At 2.37 M deployed parameters MobileNetV2 is by a factor of five the smallest of the three PyTorch networks, and it was the only one of them to show a widening train/validation gap within the fixed 15-epoch budget; the additional transforms were introduced to close that gap. Their effect on the reported comparison runs against MobileNetV2 rather than in its favour, in three respects. Heavier augmentation depresses the fit to the training distribution, which is directly visible in Figure 3 as MobileNetV2's slowest loss descent and its highest terminal training loss.

The model nevertheless records the lowest FP32 accuracy of the four at 95.43%, so the deviation cannot have produced the FP32 ordering of Table 5. And the augmentation policy is orthogonal to the central result of the paper, because post-training quantization is applied to an already-trained, frozen graph and the transforms used to obtain those weights do not enter the calibration procedure at all. We record the deviation as a real qualification on the strict-identity claim; a fully uniform augmentation policy across the three PyTorch models, with the resulting train/validation gap accepted, is a straightforward refinement for future work.

The second deviation is larger and concerns YOLOv8n-cls, which is trained through its native Ultralytics loop rather than through the shared PyTorch harness. That loop supplies its own augmentation policy — RandAugment, mosaic composition, random erasing, HSV jitter, and random scaling and translation — its own automatically selected optimizer at an initial learning rate of $1\times10-2$, its own random seed, its own[1][0] input scaling without the ImageNet mean and standard-deviation step, and its own 80/20 partition of the corpus, the last of which is treated separately under Evaluation Metrics. These are the defaults of a reference implementation that cannot be reduced to the shared harness without departing from the way the model is actually used in practice, which is the comparison a deploying engineer needs.

The consequence is that the FP32 figure for YOLOv8n-cls is not matched to the other three as tightly as they are matched to one another, and the FP32 comparison involving it should be read as indicative rather than as a controlled contrast. The quantization comparison, which is the finding of interest, is less exposed to these training-side differences, since post-training quantization operates on the exported graph rather than on the procedure that produced it.

Model Architectures and Fine-Tuning: The three PyTorch models were initialized from ImageNet-pretrained weights with early layers frozen, and in every case the 1000-class

ImageNet classifier was discarded in favour of a compact binary head (Table 3). This is why the deployed parameter counts, measured from the exported ONNX graphs, fall well below the figures usually quoted: VGG-16's replacement head alone removes roughly 123 M parameters, an 89% reduction, while leaving its feature extractor intact. YOLOv8n-cls is the exception, trained through its native Ultralytics pipeline, which was invoked with imgsz = 94.

The network is not trained at that size. The Ultralytics loader rounds a requested input size up to the next multiple of the model's maximum stride, which is 32 for YOLOv8n-cls, so 94 is resolved to 96 and the model is trained and validated at 96×96. The exported ONNX graph carries a fixed 96×96 input, which we confirmed from the shape of its input tensor, and both the ONNX Runtime evaluation path and the quantized evaluation path feed it 96×96 tensors. The effective input resolution is therefore identical to that of ResNet-18 and MobileNetV2, and no model gains or loses a preprocessing advantage from this setting: the sole difference is one additional bilinear resize step of 94 → 96 that never occurs, because the requested value is resolved before any image is read. The value 94 is an artefact of the training invocation with no effect on the deployed graph, and is stated here only so that the run is reproducible. Because the Ultralytics loop exposes no sampler hook, its class imbalance was addressed by oversampling the minority class during dataset preparation.

Table 3. Deployed architecture configurations. Inputs are square (224×224 and 96×96); parameter counts are measured from the exported ONNX graphs after head replacement.

Model	Input	Params	Unfrozen layers	Replacement classifier head
[20]	224 px	14.78 M	Conv. base retained	AdaptiveAvgPool2d(1×1) → Flatten → Linear(512,128)
[21]	96 px	11.30 M	layer4	→ ReLU → Dropout(0.5) → Linear(128,2)
[22]	96 px	2.37 M	Last 3 inverted-residual blocks	Linear(512,256) → ReLU → Dropout(0.4) → Linear(256,2)
[23]	96 px	1.44 M	Native Ultralytics fine-tuning	Linear(1280,128) → ReLU → Dropout(0.5) → Linear(128,2)

Training and Quantization Procedure:

For the PyTorch models, class imbalance was handled with a Weighted Random Sampler drawing each sample with probability inversely proportional to its class frequency, which presents an approximately balanced stream each epoch without discarding majority-class data. All models used Adam, cross-entropy loss and a fixed 15-epoch budget, with the highest-validation-accuracy checkpoint retained for export; the remaining hyperparameters are listed in Table 2.

Trained models were exported to ONNX and quantized with the ESP-PPQ framework targeting the ESP32-S3, using per-tensor symmetric INT8 for weights and activations with scales calibrated on a representative subset of the training partition. Two calibration strategies were run for every model so that their effects could be separated. The baseline is plain min/max calibration, which sets each tensor's scale from the extreme values observed over the calibration set. The reported configuration adds layerwise equalization, which rescales adjacent layers to even out per-channel weight ranges before quantization [24], and learned step-size quantization (LSQ), in which the per-tensor scales are treated as trainable parameters and optimized while the weights remain frozen [25]. LSQ as applied here therefore calibrates the quantizer, not the network; it is not full quantization-aware training, and that distinction matters for the Discussion.

The quantized graphs were emitted in ESP-DL's .espdl format and integrated into an ESP-IDF project for hardware-accelerated inference [26]. On device, faces are localized by

[27] rather than a learned detector, since its integral-image evaluation costs a small fraction of a CNN forward pass and leaves the memory budget for the classifier.

Evaluation Metrics:

Models were evaluated on the held-out 3,811-image partition of Table 1. One exception applies: YOLOv8n-cls is trained through the Ultralytics pipeline, which performs its own 80/20 partitioning, so its FP32 result is measured on that pipeline's 3,832-image split while its INT8 result, produced outside that loop, is measured on the common partition. Both splits are drawn from the same corpus and differ in size by 0.6%, but they are not identical, and every comparison involving YOLOv8n-cls carries this caveat. Because a caveat alone does not let the reader judge the size of the mismatch, the exported FP32 YOLOv8n-cls graph was additionally evaluated on the common 3,811-image partition of Table 1, through the same ONNX Runtime path and the same Ultralytics preprocessing convention used for its own split. On the common partition it attains 97.98% (95% CI 97.48–98.38), against 97.78% (97.27–98.20) on the Ultralytics split; the two intervals overlap across almost their entire width, so the two partitions are not distinguishable for this model. Measured on the single common partition, the FP32-to-INT8 difference for YOLOv8n-cls is therefore 43.40 points (97.98% → 54.58%), against the 43.20 points obtained across the two partitions; the conclusion is unchanged either way. One further disclosure is required for the same-partition figure. Because the Ultralytics loop partitions the merged corpus independently of the split in Table 1, 2,154 of the 3,811 common test images fallinside its training set. We therefore evaluated the two subgroups separately: accuracy is 97.96% on the 2,154 seen images and 98.01% on the 1,657 unseen images, a difference of 0.05 points, far smaller than the sampling uncertainty on either subgroup, so the overlap introduces no measurable optimistic bias. Both figures are reported so that the FP32–INT8 comparison for YOLOv8n-cls can be read on one partition. Even so, the drop is described throughout as an estimate and not as a strictly paired difference of the kind available for the three PyTorch models, whose FP32 and INT8 accuracies are measured on identical images.

Beyond overall accuracy, per-class precision, recall and F1 were computed for both classes, together with macro-averaged F1 and the Matthews correlation coefficient (MCC), the appropriate single-figure summary for an imbalanced binary problem because it uses all four cells of the confusion matrix. With the masked class positive and TP, FN, FP and TN the counts of correctly identified masked faces, missed masked faces, unmasked faces wrongly flagged as masked, and correctly identified unmasked faces, the per-class metrics are given by (1) and the Matthews correlation coefficient by (2):

$$\text{Precision} = TP/(TP+FP), \text{Recall} = TP/(TP+FN), F1 = 2\cdot\text{Precision}\cdot\text{Recall}/(\text{Precision}+\text{Recall}) \quad (1)$$

$$MCC = (TP\cdot TN - FP\cdot FN)/\sqrt{[(TP+FP)(TP+FN)(TN+FP)(TN+FN)]} \quad (2)$$

Every accuracy is reported with a 95% Wilson score confidence interval, preferred over the normal approximation near the boundaries of the unit interval; two models are treated as statistically distinguishable only where their intervals do not overlap. These intervals quantify sampling uncertainty on the finite test partition and nothing else. Each model was trained exactly once, from a single fixed random seed, so no interval reported in this paper carries information about seed-to-seed or run-to-run variability, and non-overlap should be read as a statement about this test set rather than as evidence that a repeated training run would reproduce the same ordering. The restriction applies to every confidence interval given below and is revisited under Limitations. Latency is the mean of 200 single-image forward passes after 20 warm-up runs. Quantization robustness is summarized by two derived quantities: accuracy retention, the ratio of INT8 to FP32 accuracy, and error inflation, the ratio of INT8 to FP32 error rate. Error inflation is the more discriminating, since an already-accurate model has little room to lose accuracy in absolute terms but can multiply its error rate

many times over.

Results:

Training Convergence:

All four models completed training within the 15-epoch budget without divergence or oscillation. Figure 3 shows the training loss trajectories and Figure 4 the validation accuracy curves for the three PyTorch models. The loss curves separate early: VGG-16, with the largest surviving feature extractor and the highest input resolution, descends fastest over the first three epochs, while MobileNetV2 — the most heavily augmented model — descends most slowly and retains the highest terminal training loss, which is the expected signature of regularization rather than of underfitting, since its validation accuracy continues to track its training accuracy closely throughout.

Validation accuracy for all three PyTorch models exceeds 93% by epoch 5 and is essentially flat after epoch 10, with the ReduceLROnPlateau schedule firing once for each model in the range of epochs 8–11. The stable validation accuracy curves in Figure 4 indicate that none of the models overfitted materially within the budget, and that a longer schedule would be unlikely to change the ranking. YOLOv8n-cls is trained through the Ultralytics loop, which does not expose a per-epoch validation accuracy in a form commensurable with the PyTorch curves; it is therefore reported only in the tabulated results.

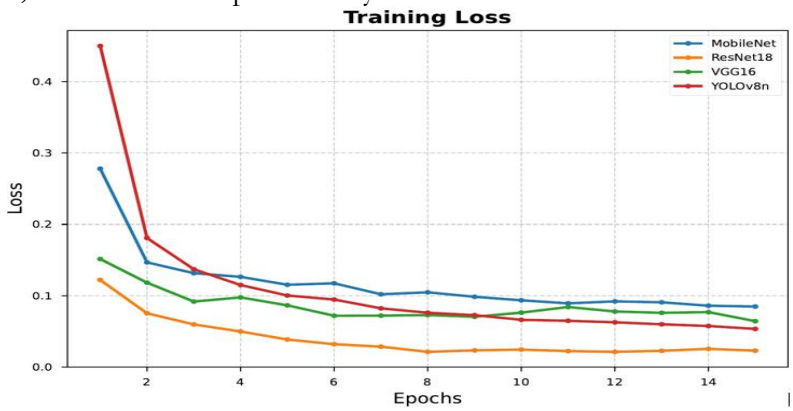

Figure 3. Training loss across 15 epochs for the three PyTorch models, showing stable convergence without divergence or oscillation. VGG-16 descends fastest over the first three epochs; MobileNetV2 descends most slowly and retains the highest terminal training loss, which is the expected signature of its heavier augmentation regime (Table 2) rather than of underfitting.

YOLOv8n-cls is trained through the Ultralytics loop and reports a loss that is not commensurable with these curves, so it is omitted.

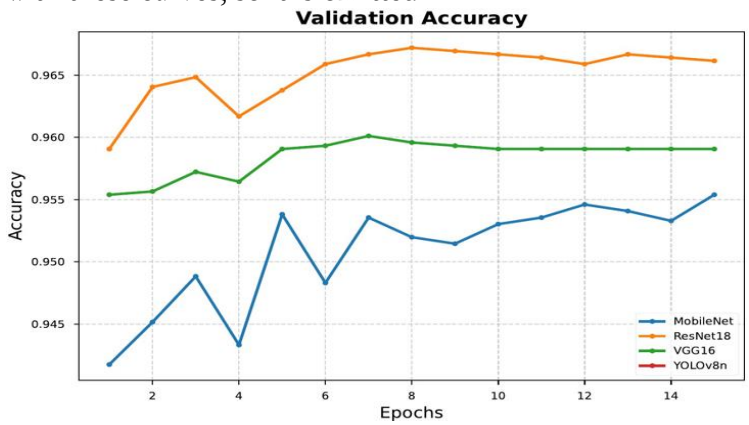

Figure 4. Validation accuracy across 15 epochs for the three PyTorch models. All three exceed 93% by epoch 5 and are essentially flat after epoch 10, indicating convergence within

the fixed budget; the ReduceLROnPlateau schedule fires once per model between epochs 8 and 11. The absence of a widening train/validation gap indicates that none of the three overfitted materially.

YOLOv8n-cls is omitted for the reason given in Figure 3. Floating-Point Classification Performance. Table 4 reports the FP32 confusion matrices and Table 5 the per-class metrics derived from them. In floating point the four models are closely matched. YOLOv8n-cls attains the highest accuracy at 97.78% (95% CI 97.27–98.20), followed by ResNet-18 at 96.72% (96.11–97.24), VGG-16 at 96.01% (95.34–96.59) and MobileNetV2 at95.43% (94.72–96.05). The confidence intervals of ResNet-18 and VGG-16 overlap, as do those of VGG-16 and MobileNetV2, so those pairs are not statistically distinguishable on this test set; YOLOv8n-cls is separated from all three, and ResNet-18 from MobileNetV2.

The more informative result is that the imbalance-handling strategy worked for every model. Per-class recall remains above 95% for both classes in all four cases (Table 5), so no model achieved its headline accuracy by quietly favoring the majority unmasked class — the failure mode that raw accuracy would conceal on a 39/61 corpus. Macro-F1 tracks accuracy closely for all four models (95.40–97.67%), and the MCC values of 0.908–0.953 confirm that the predictions are strongly associated with the ground truth across both cells of the matrix rather than driven by prior class frequency.

The error profiles differ in a way that matters operationally. The positive class throughout this paper is with_mask, as stated under Evaluation Metrics and in the note to Table 5. An unmasked face classified as masked is therefore a false positive, and a masked face classified as unmasked is a false negative. For a compliance-monitoring application the costly error is the false positive, because it silently passes a violation, whereas a false negative merely raises an alert that a human operator can dismiss. False positives are the errors that depress masked-class precision, and that is the column of Table 5 on which the models should be compared for this purpose. MobileNetV2 is the weakest model on this axis, producing 96 false positives (4.61% of the 2,083 unmasked test images) and the lowest masked-class precision of the four at 94.50%. ResNet-18 is the strongest among the PyTorch models, with 47 false positives (2.26%) and masked-class precision of 97.23%; VGG-16 lies between them, with 83 (3.99%) and 95.24%. YOLOv8n-cls produces the most balanced profile overall on its own split, with 39 false positives against 46 false negatives, an unmasked-class recall of 98.33%, and the highest MCC at 0.9534.

Table 4. Confusion matrices for all four models in FP32 precision.

Model	Actual class	Predicted: Mask	Predicted: No Mask
VGG-16	Mask	1,659	69
	No Mask	83	2,000
ResNet-18	Mask	1,650	78
	No Mask	47	2,036
MobileNetV2	Mask	1,650	78
	No Mask	96	1,987
YOLOv8n-cls*	Mask	1,450	46
	No Mask	39	2,297

*YOLOv8n-cls was evaluated on its own 80/20 split of the merged corpus (3,832 images) generated by the Ultralytics dataset preparation; the three PyTorch models share the common 3,811-image test partition of Table 1.

All values in percent except MCC. Positive class is with_mask. Figure 5 presents the four confusion matrices in graphical form, which makes the error asymmetries directly comparable. Diagonal dominance is evident in all four panels; the off-diagonal structure is what separates the models. ResNet-18 and YOLOv8n-cls concentrate their residual error on

false negatives, meaning that they occasionally miss a mask but rarely report one that is not present, whereas MobileNetV2 shows the reverse tendency and VGG-16 lies between them. For deployment purposes the ResNet-18 profile is the most desirable of the three PyTorch models, because a false negative produces a spurious non-compliance alert that a human operator can dismiss, whereas a false positive produces a silent false clearance.

Table 5. Per-class and aggregate classification metrics in FP32 precision, derived from Table 4. Accuracy is reported with a 95% Wilson score confidence interval.

Model	Accuracy (95% CI)	Prec. Mask	Recall Mask	F1 Mask	Prec. No Mask	Recall No Mask	F1 No Mask	Macro-F1
VGG-16	96.01 (95.34–96.59)	95.24	96.01	95.62	96.67	96.02	96.34	95.98
ResNet-18	96.72 (96.11–97.24)	97.23	95.49	96.35	96.31	97.74	97.02	96.69
MobileNetV2	95.43 (94.72–96.05)	94.50	95.49	94.99	96.22	95.39	95.81	95.40
YOLOv8ncls	97.78 (97.27–98.20)	97.38	96.93	97.15	98.04	98.33	98.18	97.67

Effect of Quantization on Model Size:

Table 6 reports the memory footprint before and after INT8 conversion. Two distinct quantities are separated there, because they answer different questions and were previously conflated. The first is the theoretical parameter-memory reduction, which follows arithmetically from storing each parameter in one byte rather than four and is exactly 75.00% for every model.

The second is the reduction measured between the two exported artefacts actually written to disk — the FP32 ONNX graph and the INT8 .espdl graph — which is the number that governs whether a model fits the device. Computed from those two comparable quantities, the measured reduction is 73.45–74.97% across architectures. The shortfall against the 75.00% ceiling is the metadata, graph structure and quantization parameters carried by the containers, which are proportionally larger for the smaller graphs: 1.55 points for MobileNetV2 and 1.16 for YOLOv8n-cls, against 0.03 and 0.11 for VGG-16 and ResNet-18.

Quantization is therefore doing exactly what is expected of it in compression terms, independently of architecture— which is the point of comparison against the accuracy results that follow.

In absolute terms the reduction changes the deployability of only two of the four models. MobileNetV2 falls from 9.50 MB to 2.52 MB and YOLOv8n-cls from 5.77 MB to 1.51 MB, both comfortably inside the 8 MB PSRAM of the ESP32-S3-CAM. ResNet-18 (11.35 MB) and VGG-16 (14.81 MB) remain above the ceiling even after quantization, and no amount of INT8 conversion alone brings them under it.

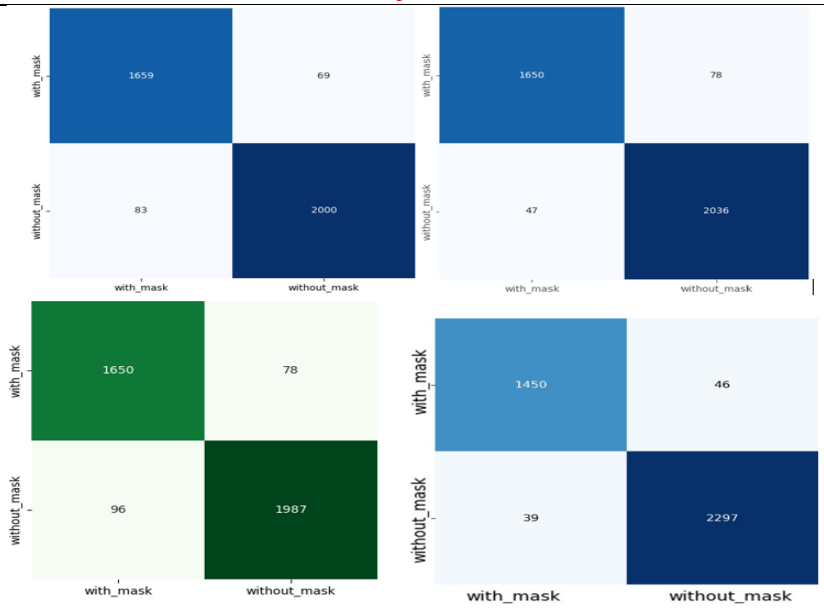

Figure 5. FP32 confusion matrices: VGG-16 (top left), ResNet-18 (top right), MobileNetV2 (bottom left) and YOLOv8n-cls (bottom right). Rows are the true class and columns the predicted class, with counts as tabulated in Table 4; the upper-right cell of each panel is the false-negative count and the lower-left cell the false-positive count, with_mask as the positive class. The three PyTorch panels are computed on the common 3,811-image partition, the YOLOv8n-cls panel on that model's own 3,832-image Ultralytics split.

Table 6. Model size reduction through INT8 quantization, separating the theoretical parameter-memory reduction from the reduction measured between the exported FP32 and INT8 graphs. Throughout this paper 1 MB = 106 bytes.

Model	**Params**	**Param mem., FP32**	**Param mem., INT8**	**Theoretical reduction**	**Exported FP32 graph**	**Exported INT8 *.espdl***	**Measured reduction**
VGG-16	14.78 M	59.12 MB	14.78 MB	75.00%	59.18 MB	14.81 MB	74.97%
ResNet-18	11.30 M	45.21 MB	11.30 MB	75.00%	45.22 MB	11.35 MB	74.89%
MobileNetV2	2.37 M	9.48 MB	2.37 MB	75.00%	9.50 MB	2.52 MB	73.45%
YOLOv8ncls	1.44 M	5.75 MB	1.44 MB	75.00%	5.77 MB	1.51 MB	73.84%

1 MB = 106 bytes. Parameter memory is the deployed parameter count of Table 3 at 4 bytes (FP32) and 1 byte (INT8) each; the theoretical reduction follows from those two columns alone and is 75.00% by construction. Exported sizes are on-disk file sizes: the ONNX graph for FP32 and the ESP-DL. espdl graph for INT8. The measured reduction is computed from the two exported sizes, which are comparable quantities, and falls below 75.00% by the container overhead. The deployability assessment in Tables 7 and 10 uses the exported INT8 size, since that is what must be resident on the device.

Effect of Quantization on Accuracy:

Where compression was uniform, accuracy retention was not. Table 7 sets the FP32 and INT8 accuracies side by side. VGG-16 loses 0.67 percentage points (96.01% to 95.34%); the two confidence intervals overlap across most of their width, so this change is at the edge of what a test set of this size can resolve and is best described as no material loss. ResNet-18

loses 10.44 points and MobileNetV2 14.30, both far outside their intervals and unambiguous. YOLOv8n-cls loses 43.20 points, falling to 54.58%.

The nature of the YOLOv8n-cls failure is worth stating precisely, because 54.58% could in principle describe a merely noisy classifier. It does not. On the 3,811-image common partition the majority-class (unmasked) rate is 54.66%, so the quantized model is marginally below the accuracy of a constant predictor that never reports a mask at all. Inspection of its predictions confirms the reason: of the 1,728 masked faces in the partition, only 9 are recovered, a masked-class recall of 0.52%, while essentially the whole unmasked class is retained. The quantized network has not degraded gracefully; it has ceased to respond to the masked class altogether, and its residual accuracy is an artefact of the class prior rather than evidence of any retained discriminative capacity. For a compliance-monitoring system this is the worst possible failure mode, since the model reports universal compliance and would raise no alarm while doing so.

Table 7. Classification accuracy before and after INT8 quantization, with 95% Wilson score confidence intervals.

Model	FP32 accuracy (95% CI)	INT8 accuracy (95% CI)	Accuracy drops	Deployable on ESP32-S3
VGG-16†	96.01(95.3496.5)	95.34(94.6195.95)	0.67	No (14.81 MB)
ResNet-18	96.72(96.1197.2)	86.28(85.1587.33)	10.44	No (11.35 MB)
MobileNetV2	95.43(94.7296.0)	81.13(79.8682.34)	14.30	Yes
YOLOv8ncl‡	97.78(97.2798.2)	54.58(52.9956.15)	43.20	Yes

All values in percent. †The VGG-16 INT8 figure was measured with the ONNX Runtime INT8 build rather than the ESP-DL/LSQ path used for the other three models, because the 14.81 MB graph could not be deployed to the target device. ‡The YOLOv8n-cls FP32 figure is measured on the 3,832-image Ultralytics split and its INT8 figure on the common 3,811-image partition; see "Evaluation Metrics".

The ranking inverts completely across the FP32–INT8 boundary. In floating point, the order is YOLOv8n-cls > ResNet-18 > VGG-16 > MobileNetV2; after quantization it is VGG-16 > ResNet-18 > MobileNetV2 > YOLOv8n-cls. The best floating-point model becomes the worst quantized model by a margin of 40.76 points, and across these four models the rank correlation between FP32 and INT8 accuracy is negative.

Quantization Robustness:

Because the four models do not start from the same FP32 accuracy, absolute point losses understate the differences between them. Table 8 therefore reports accuracy retention and error inflation alongside the raw drop. On this view the separation is far wider than Table 7 suggests. VGG-16 retains 99.30% of its floating-point accuracy and inflates its error rate by a factor of only 1.17, meaning INT8 conversion adds less than a fifth again to the errors it was already making. ResNet-18 and MobileNetV2 form a middle group that is remarkably consistent despite their fivefold difference in size, inflating error by 4.18× and 4.13× respectively and retaining 89.21% and 85.02% of baseline accuracy. YOLOv8n-cls inflates its error rate by 20.46× and retains 55.82%.

The near-identical error inflation of ResNet-18 and MobileNetV2 is itself informative. These two models differ substantially in parameter count (11.30 M against 2.37 M), in width, and in whether they use depth wise convolution at all, yet they sustain almost exactly the same proportional damage. For this pair, therefore, the quantization penalty does not track parameter count, and the intuition that smaller models must quantize worse because they carry less redundancy is not supported by these two observations. We offer the alternative reading — that what governs the penalty is whether an architecture contains structures that a single

per-tensor scale cannot represent, of which VGG-16 has none, YOLOv8n-cls several, and the two intermediate models some — as an interpretation consistent with the four measurements reported here and with the published quantization literature [24][16], and not as an established general result. Four architectures evaluated on one task cannot separate capacity from architectural composition, because the two are confounded across a sample of this size; establishing the point would require a controlled sweep in which width and depth are varied while the operator set is held fixed, which is beyond the scope of this study.

The final column of Table 8 recasts the same data as accuracy per megabyte of deployed model, which is the quantity an engineer working against a fixed memory budget actually optimizes. Here the ordering changes again: MobileNetV2 delivers 32.19 accuracy points per megabyte and YOLOv8n-cls 36.15, against 6.44 for VGG-16 and 7.60 for ResNet-18. The figure for YOLOv8n-cls is not meaningful given that the underlying model does not work, which illustrates why efficiency ratios of this kind must always be read against an absolute accuracy floor rather than on their own.

Table 8. Quantization robustness by architecture. Error inflation is the ratio of INT8 to FP32 error rate; accuracy retention is INT8 accuracy as a percentage of FP32 accuracy.

Model	FP32 error	INT8 error	Error inflation	Accuracy retention	Accuracy per MB (INT8)
VGG-16	3.99%	4.66%	1.17×	99.30%	6.44
ResNet-18	3.28%	13.72%	4.18×	89.21%	7.60
MobileNetV2	4.57%	18.87%	4.13×	85.02%	32.19
YOLOv8n-cls	2.22%	45.42%	20.46×	55.82%	36.15

Accuracy per MB is INT8 accuracy divided by INT8 model size from Table 6; for YOLOv8n-cls the figure is reported for completeness only, since the model is non-functional at that accuracy.

Calibration Ablation:

The INT8 figures in Table 7 are obtained after layer-wise equalization and LSQ scale optimization; they are the best results achieved, not the naive ones. Table 9 separates the two calibration strategies for the two models where the refinement matters most. Plain min/max calibration reduces MobileNetV2 to 75.86%, which equalization plus LSQ recovers to 81.13% — a gain of 5.27 points, or roughly 37% of the total quantization penalty. For YOLOv8n-cls the same treatment moves accuracy from 45.30% to 54.58%, a nominal gain of 9.28 points that is nonetheless meaningless in practice, since both figures describe a model that has collapsed onto one class. At 45.30% the plain-calibrated model sits almost ten points below the majority-class baseline, which is to say its residual output is weakly anti-correlated with the labels.

The ablation therefore establishes two things. Scale-level refinement produces a real and reproducible improvement, so the losses reported in Table 7 are not an artefact of careless calibration and cannot be dismissed as such. But it does not alter the ordering of the architectures, and it does not rescue any model that plain calibration breaks. The vulnerability being measured is therefore a property of the architecture rather than of the quantization procedure applied to it.

All values in percent. The majority-class rate on the evaluation partition is 54.66%, so neither INT8 configuration of YOLOv8n-cls is usable.

Table 9. Calibration ablation: INT8 accuracy under plain min/max calibration versus layerwise equalization combined with LSQ scale optimization.

Model	FP32	INT8, min/max	INT8, equalization + LSQ	Recovery
MobileNetV2	95.43	75.86	81.13	+5.27
YOLOv8n-cls	97.78	45.30	54.58	+9.28

Inference Latency and Deployment. Table 10 reports latency on both platforms. On the desktop CPU all four FP32 models complete a single face in 1.12–3.11 ms, so on that platform classification is negligible against frame capture and Haar-cascade localization, and latency provides no basis for choosing between the architectures. The measurements are reported chiefly as a reference point against which the on-device figures can be read.

On the ESP32-S3-CAM the picture is entirely different. MobileNetV2 executes in 576.03 ms per face, giving roughly 1.7 classifications per second, while YOLOv8n-cls executes in 242.20 ms, or about 4.1 per second — a 2.4× advantage consistent with its smaller graph, though one purchased at the cost of an unusable model.

The slowdown relative to the desktop CPU is 324× for MobileNetV2 and 141× for YOLOv8n-cls. Two points follow. First, the magnitude of the gap reflects the absence of a floating-point unit, a SIMD width of a few lanes rather than hundreds, and memory bandwidth measured in tens rather than thousands of megabytes per second; desktop-simulated INT8 benchmarks therefore cannot be used to estimate microcontroller latency even to within an order of magnitude. Second, the gap is not uniform across architectures — MobileNetV2 loses more than twice as much ground as YOLOv8n-cls — because its depth wise convolutions have a low arithmetic intensity that leaves the device memory-bound, whereas the desktop cache hierarchy hides that cost entirely. Relative latency measured off-target is thus not a reliable predictor of relative latency on-target.

VGG-16 and ResNet-18 were not executed on the device at all. At 14.81 MB and 11.35 MB their quantized graphs exceed the board's 8 MB of external PSRAM, so the exclusion is a capacity constraint and not a failure of the quantization pipeline; both quantized successfully and both run correctly under ONNX Runtime. This is the practical significance of the memory result in Table 6: VGG-16 is simultaneously the most quantization-robust model tested and one that cannot be placed on the target hardware.

Table 10. Inference latency per face on desktop CPU (FP32) and on the ESP32-S3-CAM (INT8).

Model	Desktop CPU, FP32	ESP32-S3, INT8	Throughput on device	Deployed	Throughput on device
VGG-16	3.11 ms	—	—	No (exceeds 8 MB PSRAM)	—
ResNet-18	1.12 ms	—	—	No (exceeds 8 MB PSRAM)	—
MobileNetV2	1.78 ms	576.03 ms	1.7 FPS	Yes	1.7 FPS
YOLOv8n-cls	1.72 ms	242.20 ms	4.1 FPS	Yes	4.1 FPS

Desktop latency is the mean of 200 single-image ONNX Runtime inferences after 20 warm-up runs, measured on the machine of Table 2. Dashes indicate models that exceed the board's memory and were therefore never executed on device.

Discussion:

Why Quantization Robustness Diverges from Floating-Point Accuracy:

The central result of this study is the inversion between Tables 5 and 7, and it has a coherent mechanistic explanation rooted in how per-tensor INT8 quantization interacts with each architecture.

VGG-16's robustness follows from the uniformity of its design. A stack of 3×3 convolutions separated by ReLU and max-pooling produces activation distributions that are

similar in scale from layer to layer and bounded below at zero, so a single per-tensor scale represents each tensor with little wasted range. There are no depth wise layers whose channels carry independent magnitudes, and no unbounded activation function to generate outliers that stretch the quantization grid. The 0.67-point loss observed here is consistent with the fraction-of-a-point losses reported for plain convolutional stacks in the general quantization literature [11][12], and indicates that VGG-16's INT8 behavior is essentially the textbook case.

MobileNetV2's degradation has a well-characterized cause. Its depth wise separable convolutions apply one filter per input channel, and the resulting per-channel weight magnitudes vary over orders of magnitude within a single tensor. A per-tensor scale must accommodate the largest channel, which leaves the small-magnitude channels represented by a handful of quantization levels and effectively destroys their contribution. This is precisely the failure mode that layer-wise equalization was designed to address [24], and the ablation in Table 9 is consistent with that mechanism: rescaling adjacent layers to even out the ranges recovers 5.27 points. The mechanism itself is established in the quantization literature rather than demonstrated here — depth wise-separable networks are documented as the canonical difficult case for per-tensor INT8 [16] — and what the present ablation adds is only that the remedy designed for that failure mode produces a real but partial recovery on this model. That is evidence consistent with the explanation; it does not on its own exclude other contributions, and we present it as an interpretation rather than as a demonstrated cause. That it recovers only about a third of the penalty reflects the limits of per-tensor quantization on this architecture; per-channel weight quantization, which the ESP-DL target does not expose for all operator types, would be the natural remedy.

YOLOv8n-cls presents a qualitatively different failure. Its collapse is not graceful degradation but a loss of class response, and it persists through LSQ. We propose an explanation in terms of two architectural features, and state at the outset that it is a hypothesis: no ablation isolating either feature was performed in this study, and what follows is a mechanism consistent with the observation rather than a demonstrated cause. The SiLU activation, $x \cdot \sigma(x)$, is unbounded above and non-monotonic near zero, and would on that basis be expected to produce long-tailed activation distributions in which rare large values dictate the per-tensor scale while the dense, informative region near zero is left with few quantization levels. Published work on quantization outliers reports precisely this sensitivity in general terms, finding that networks whose activation distributions carry heavy tails degrade severely at low precision unless the outliers are treated explicitly[28]; that work does not examine YOLOv8n-cls, so it establishes the plausibility of the mechanism and not its operation in this case. The CSPDarknet backbone may compound this by splitting and concatenating feature maps across stages, so that tensors with substantially different statistics are joined and quantized under a shared scale. If the error introduced at these points exceeds the margin separating the two class logits, the argmax becomes constant — which is the signature observed, 9 of 1,728 masked faces recovered.

We emphasize that the observation is also consistent with other mechanisms, including numerical issues confined to a small number of layers, and does not by itself identify the responsible structure.

Three ablations would test this account directly, and we name them here as the necessary next step rather than presenting the question as settled. First, substituting ReLU6 for SiLU throughout the backbone and retraining under an otherwise unchanged protocol would isolate the activation-range contribution: if the activation is responsible, the ReLU6 variant should quantize with a penalty of the order of VGG-16's rather than of the one observed. Second, a per-layer sensitivity sweep, in which one layer at a time is quantized while the remainder are held in FP32, would localize the collapse to specific tensors and establish whether the concatenation points are among them. Third, recording per-tensor activation

histograms and their kurtosis at calibration time for all four models would test the distributional premise directly, by establishing whether YOLOv8n-cls's tensors are in fact more heavy-tailed than those of the models that survive quantization. None of the three was run here, and until they are the explanation above stands as a hypothesis.

That LSQ fails to repair this is diagnostic rather than disappointing. LSQ optimizes the quantizer's step sizes while holding the weights fixed [25], so it can only find the best integer grid for a given set of floating-point weights. If no such grid preserves the class margin, the weights themselves must move. Recovering YOLOv8n-cls would therefore be likely to require full quantization-aware training — forward passes through simulated quantizers with gradients propagated to the weights— or architectural substitution of ReLU6 for SiLU before training.

This is the clearest practical lesson of the study: the standard advice to reach for post-training quantization first, and escalate to QAT only if needed, silently assumes that the escalation is available, and for some architectures the gap between the two is the difference between a working and a non-existent model.

Implications for Architecture Selection:

Read jointly, Tables 6, 7 and 10 define a selection problem with no clean solution on this hardware. VGG-16 gives the best quantized accuracy and cannot be deployed. YOLOv8n-cls gives the best latency and best FP32 accuracy and is unusable after quantization. ResNet-18 is the best-balanced model on paper — a moderate 10.44-point loss to a still-serviceable 86.28% — and also exceeds the memory ceiling. MobileNetV2 is the only model that clears every hard constraint, and it does so at 81.13% accuracy and 1.7 FPS, which is functional for periodic compliance sampling but not for tracking individuals through a busy doorway.

The practical recommendation follows the deployment context rather than a single ranking. Where a gateway device, single-board computer or phone-class SoC is available, quantized VGG-16 at 95.34% is the strongest option tested and its 14.81 MB footprint is unremarkable on such hardware. Where inference must run on the microcontroller itself, MobileNetV2 is currently the only viable candidate among the four, and its accuracy deficit should be treated as the cost of that constraint rather than as a property of the architecture, since 14.30 of the missing 18.87 points are attributable to quantization rather than to the model.

More generally, the results argue that architecture selection for edge deployment should be conducted in the quantized regime from the outset. Selecting on FP32 accuracy and quantizing afterwards would have chosen YOLOv8n-cls here, and would have produced a system that fails completely at the last step of the pipeline — after training, export, calibration and integration work had all been done. The negative correlation between FP32 and INT8 accuracy across these four models, while measured on only four points and not to be over-generalized, is sufficient to show that FP32 accuracy cannot be assumed to be a safe proxy.

Comparison with Related Work:

The FP32 accuracies reported here are consistent with the published face-mask literature. Recent lightweight-model frameworks report accuracies in the 95–98% band on comparable two- and three-class corpora [2][3],transformer-based approaches report similar figures [4][29], and improved YOLOv8 variants report competitive detection performance [5]; the 95.43–97.78% range of Table 5 sits squarely within this envelope, which supports the validity of the training protocol. Earlier pandemic-era systems fall in the same band. The hybrid model of [30] pairs a deep transfer-learned feature extractor with classical machine-learning decision stages and reports accuracies in the mid-90s across three public mask corpora, while SSDMNV2 [31] couples a single-shot multibox detector to a MobileNetV2 classification head and reports 92.6% at real-time frame rates. Both are directly comparable to the present work in the floating-point regime, and both sit inside the envelope of Table 5.

Neither reports post-quantization accuracy, and neither deploys microcontroller-class hardware, which is the respect in which the present study extends them.

What distinguishes the present results is the quantized regime, which most of this literature does not report. Reviews of lightweight face-mask models for edge environments identify the absence of matched post-quantization evaluation as a standing gap[1] and the TinyML surveys note that quantization sensitivity is architecture-dependent without generally quantifying it for specific deployment targets [6][7][8][32][10]. The work closest to ours in intent is that of [17], who prune and quantize a YOLO detector for low-end devices and report the accuracy cost of doing so; that study examines one model family under compression rather than comparing architectures, which is the axis the present work adds.

Recent ESP32-class deployment studies likewise report per-model results on microcontroller hardware without a matched cross-architecture quantization comparison [33][34]. Studies that do deploy typically target mobile SoCs or single-board computers, where the memory ceiling that excluded VGG-16 and ResNet-18 here simply does not bind, and where floating-point execution remains available as a fallback [35]. The contribution of this work is to close that loop on genuinely microcontroller-class hardware, where both constraints bind simultaneously.

Limitations:

Several limitations bound the conclusions. The comparison rests on four architectures and one task, so the negative FP32–INT8 correlation reported above should be read as an existence proof that the two can diverge sharply, not as an estimate of a general relationship. The VGG-16 INT8 figure was obtained through ONNX Runtime rather than the ESP-DL and LSQ path used for the other three models, because the graph could not be placed on the device; the two paths implement the same per-tensor symmetric scheme, but the figure is not strictly like-for-like with the rest of Table 7 and is marked accordingly. The YOLOv8n-cls evaluation is split across two partitions — FP32 on the 3,832-image Ultralytics split, INT8 on the common 3,811-image partition — because its native pipeline performs its own dataset preparation while the quantized evaluation runs outside that loop. Both partitions are drawn from the same merged corpus and differ in size by 0.6%, so the 43.20-point drop is far too large to be an artefact of the mismatch, but the figure is not a strictly paired measurement and is best read as an estimate.

Three further constraints apply to the deployed system. Face localization uses a Haar cascade, which is fast and memory-frugal but degrades at extreme pose and in low light, so end-to-end field accuracy will fall below the classifier accuracies reported here, which are measured on pre-cropped faces. The corpus, though large, is drawn from two public collections and has not been validated against imagery from the intended deployment environment, where camera height, lens, illumination and mask types will differ. Finally, every model was trained exactly once, from a single fixed random seed, and no accuracy reported in this paper is an average over repeated runs.

The 95% Wilson intervals quantify sampling uncertainty on the finite test partition and nothing else: they do not bound seed-to-seed variability, initialization variance, or the effect of the non-deterministic elements of the training pipeline. Two consequences follow, and we state them plainly rather than leaving them to be inferred. Where two models' intervals do not overlap — ResNet-18 against MobileNetV2 in FP32, for instance — the separation is established on this test set but is not established as reproducible across training runs. And the FP32 ordering of the four models, whose full spread is only 2.35 points, lies well within the range that seed variation alone is capable of producing for models of this size, so that ordering should be read as provisional. The quantization findings are not exposed in the same degree, since the FP32-to-INT8 differences of 10.44, 14.30 and 43.20 points are an order of magnitude larger than any plausible seed effect; but confirming even those, and attaching a variance

estimate to the FP32 comparison, would require the whole experiment repeated over several seeds with accuracies reported as means and standard deviations. We identify that repetition as the single most valuable extension of this work, and the FP32 ranking of Table 5 should not be relied on for model selection until it has been carried out.

Future Work:

Repeating the entire protocol over several random seeds, and reporting every accuracy as a mean with a standard deviation across runs, is the first priority, since it is what would convert the FP32 ordering of Table 5 from a single-run observation into a supported claim. The most direct methodological extension is full quantization-aware training for MobileNetV2 and YOLOv8n-cls, which the LSQ results indicate is the necessary escalation rather than an optional refinement. Substituting ReLU6 for SiLU in the YOLOv8n-cls backbone before training would test the activation-range hypothesis advanced above directly. Per-channel weight quantization, where the target runtime supports it, should recover a further portion of MobileNetV2's deficit. On the deployment side, replacing the Haar cascade with a quantized lightweight detector would allow end-to-end field evaluation, and structured pruning or knowledge distillation applied to ResNet-18 would test whether the best-balanced architecture can be compressed under the 8 MB ceiling without surrendering its quantization robustness.

Conclusion:

This study compared VGG-16, ResNet-18, MobileNetV2 and YOLOv8n-cls for face mask classification under a matched training, quantization and deployment protocol, and carried the comparison through to physical execution on an ESP32-S3-CAM microcontroller. In floating-point precision the four models were closely matched at 95.43–97.78% accuracy with macro-F1 above 0.95 and per-class recall above 95% throughout, confirming that the task is not architecture-limited at full precision. INT8 quantization compressed all four models uniformly by 73–75%, but degraded them by 0.67, 10.44, 14.30 and 43.20 percentage points respectively — a spread that inverted the ranking entirely and left the most accurate floating-point model, YOLOv8n-cls, collapsed onto a single class at 54.58%. A calibration ablation established that layer-wise equalization and LSQ recover a real but partial share of this loss (5.27 points for MobileNetV2, 9.28 for YOLOv8n-cls) without changing the ordering, which places the vulnerability in the architectures tested rather than in the quantization procedure applied to them; on the common test partition the same FP32 YOLOv8n-cls graph attains 97.98%, so the drop is 43.40 points when measured on a single partition. Of the four, only MobileNetV2 satisfied both the 8 MB memory ceiling and a usable accuracy level, running at 81.13% and 576 ms per face; VGG-16, the most quantization-robust model tested, was excluded by memory alone. The practical implication is that architectures intended for microcontroller deployment should be selected and evaluated in the quantized regime from the outset, because floating-point accuracy proved, across the four architectures tested here, an unreliable proxy for deployed accuracy. That finding rests on four architectures, one task and a single training run per model, and is offered as an existence proof that the two can diverge sharply rather than as an estimate of a general relationship.

Acknowledgement:

The authors acknowledge the institutional support provided by the Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Pakistan, and thank the anonymous reviewers for their comments on the manuscript.

Author's Contribution:

A.A. conceived the study, implemented the training, quantization and embedded deployment pipelines, conducted all experiments, and drafted the manuscript. S.S. contributed to dataset preparation, ran the calibration ablation, and assisted with analysis of the results. A.M.A. supervised the work, contributed to the study design and to interpretation of the

results, and critically revised the manuscript. All authors reviewed and approved the final version.

Conflict of Interest:

The authors declare that there exists no conflict of interest for publishing this manuscript in IJIST. Project details. This research was not conducted as part of a funded project and received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability:

The datasets analysed are publicly available: the MAFA masked-face dataset [18] and WIDER FACE [19]. The training, quantization and evaluation code supporting the findings of this study is available from the corresponding author upon reasonable request. The study used public benchmark imagery only and involved no recruitment of human participants or collection of new personal data by the authors.

References:

[1] S. Rasheed, "Lightweight Deep Learning Models for Face Mask Detection in Real-Time Edge Environments: A Review and Future Research Directions," *Mach. Learn. Knowl. Extr. 2026, Vol. 8, Page 102*, vol. 8, no. 4, p. 102, Apr. 2026, doi: 10.3390/MAKE8040102.

[2] B. E, M. P, S. M, and O. J, "A multi-class framework for face mask compliance detection using lightweight deep learning models," *Sci. Reports 2026 161*, vol. 16, no. 1, pp. 22059-, May 2026, doi: 10.1038/s41598-026-50603-1.

[3] C. Pillay, S. Joseph, and B. van Niekerk, "Development of a Face Mask Detection and Masked Facial Recognition Model Based on a Hybrid Convolutional Neural Network," *IET Image Process.*, vol. 19, no. 1, Jan. 2025, doi: 10.1049/IPR2.70239.

[4] Y. Mao, Y. Lv, G. Zhang, and X. Gui, "Exploring Transformer for Face Mask Detection," *IEEE Access*, vol. 12, pp. 118377–118388, 2024, doi: 10.1109/ACCESS.2024.3449802.

[5] X. Ren and X. Liu, "Mask wearing detection based on YOLOv3," *J. Phys. Conf. Ser.*, vol. 1678, no. 1, p. 012089, Nov. 2020, doi: 10.1088/1742-6596/1678/1/012089.

[6] S. Somvanshi *et al.*, "From Tiny Machine Learning to Tiny Deep Learning: A Survey," *ACM Comput. Surv.*, vol. 58, no. 7, Jun. 2025, doi: 10.1145/3776588.

[7] S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," *Sensors 2025, Vol. 25, Page 3191*, vol. 25, no. 10, p. 3191, May 2025, doi: 10.3390/S25103191.

[8] M. Echchidmi and A. Bouayad, "TinyML for Sustainable Edge Intelligence: Practical Optimization Under Extreme Resource Constraints," *Technol. 2026, Vol. 14, Page 215*, vol. 14, no. 4, p. 215, Apr. 2026, doi: 10.3390/TECHNOLOGIES14040215.

[9] S. H. Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, "MCUNet: Tiny Deep Learning on IoT Devices," *NeurIPS 2020*, doi: https://doi.org/10.48550/arXiv.2007.10319.

[10] L. C. S. C. F. Capogrosso, "A machine learning-oriented survey on tiny machine learning," *researchgate*, doi: DOI:10.1109/ACCESS.2024.3365349.

[11] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2704–2713, Dec. 2017, doi: 10.1109/CVPR.2018.00286.

[12] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," *Low-Power Comput.*

Vis., pp. 291–326, Mar. 2021, doi: 10.1201/9781003162810-13.

[13] M. F. A. A. B. Nagel, "A white paper on neural network quantization," *researchgate*, 2021, doi: DOI:10.48550/arXiv.2106.08295.

[14] Raghuraman Krishnamoorthi., "Quantizing deep convolutional networks for efficient inference: A whitepaper," *researchgate*, 2018, doi: DOI:10.48550/arXiv.1806.08342.

[15] O. E. F. E. Krestinskaya, "Quantized convolutional neural networks: a hardware perspective," *researchgate*, 2025, doi: DOI:10.3389/felec.2025.1469802.

[16] A. W. Stone Yun, "Do all MobileNets quantize poorly? Gaining insights into the effect of quantization on depth wise separable convolutional networks through the eyes of multi-scale distributional dynamics," *Accept. Publ. Mob. AI*, 2021, doi: https://doi.org/10.48550/arXiv.2104.11849 Focus to learn more.

[17] G. Santacatterina and B. Z. Liberatori, "YOLO-based face mask detection on low-end devices using pruning and quantization," *researchgate*, 2022, doi: DOI:10.23919/MIPRO55190.2022.9803406.

[18] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with LLE-CNNs," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 426–434, Nov. 2017, doi: 10.1109/CVPR.2017.53.

[19] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 5525–5533, Nov. 2015, doi: 10.1109/CVPR.2016.596.

[20] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014, Accessed: Aug. 29, 2026. [Online]. Available: https://arxiv.org/pdf/1409.1556

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2016, doi: 10.1109/CVPR.2016.90.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, Jan. 2018, doi: 10.1109/CVPR.2018.00474.

[23] Ultralytics, "Ultralytics YOLOv8 Documentation," *Ultralytics*, 2025, [Online]. Available: https://docs.ultralytics.com/

[24] M. Nagel, M. Van Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1325–1334, Jun. 2019, doi: 10.1109/ICCV.2019.00141.

[25] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned Step Size Quantization," *8th Int. Conf. Learn. Represent. ICLR 2020*, Feb. 2019, Accessed: Aug. 29, 2026. [Online]. Available: https://arxiv.org/pdf/1902.08153

[26] "ESP-DL: Espressif deep-learning library for AIoT applications," 2025, [Online]. Available: https://github.com/espressif/esp-dl

[27] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, 2001, doi: 10.1109/CVPR.2001.990517.

[28] G. Yoo, J.; Ban, "Efficient deep learning model compression for sensor-based vision systems via outlier-aware quantization," *Sensors*, 2025, doi: https://doi.org/10.3390/s25092918.

[29] B. Pandya, D. Patel, and K. C. Yow, "Face Mask Detection using Vision

Transformer," *Can. Conf. Electr. Comput. Eng.*, vol. 2023-September, pp. 268–272, 2023, doi: 10.1109/CCECE58730.2023.10288856.

[30] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement*, vol. 167, p. 108288, Jan. 2021, doi: 10.1016/J.MEASUREMENT.2020.108288.

[31] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," *Sustain. Cities Soc.*, vol. 66, p. 102692, Mar. 2021, doi: 10.1016/J.SCS.2020.102692.

[32] H. A. D. V. J. P. I. A. Abushahla, "Neural network quantization for efficient deployment on resource-constrained microcontrollers: A survey," *ResearchGate.,* 2025, doi: DOI:10.48550/arXiv.2508.15008.

[33] F. Ben Aicha., "Optimized embedded artificial intelligence: Implementation of convolutional neural networks on the ESP32-CAM,"," *Computing*, doi: doi: 10.1007/s00607-025-01559-z.

[34] R. Chen, P. Wang, B. Lin, B. Lin, or hasn't claimed this research yet. This person is not on ResearchGate, and L. Wang, "An optimized lightweight real-time detection network model for IoT embedded devices," *Sci. Rep.*, 2025, doi: DOI:10.1038/s41598-025-88439-w.

[35] C. M. Bhushan *et al.*, "Deploying TinyML for energy-efficient object detection and communication in low-power edge AI systems," *Sci. Reports 2025 151*, vol. 15, no. 1, pp. 44299-, Dec. 2025, doi: 10.1038/s41598-025-27818-9.

Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.