Copenhagen Business School

MSc. Business Administration and Data Science

KAN-CDSCO2004U Machine Learning and Deep Learning

Banana Ripeness Classification

A benchmark of pretrained vs. from-scratch CNN architectures

Mikolaj Sapek: S185880
Julia Nowak : S160787
Yasemin Pagano: S185872
Teodor Nedevski : S185921
Supervisor: Somnath Mazumdar
Submission date: Spring 2026

Number of pages: 14

Characters: 31388

Table of Contents

Abstract

Food waste problems could be made worse by supply chain disruptions brought on by recent political unrest, wars, conflicts, and a decline in the international rule of law, all of which have an impact on supply chain predictability. Motivated by this shift in stability in international trade, as well as a general objective to reduce food waste, in this paper, we explored the use of deep learning image classifiers, specifically using a pre-trained model, MobileNetV2, with transfer learning as a benchmark; a CNN trained from scratch on the same task dataset; and a CNN with custom residuals and SE attention. The dataset is a publicly accessible set of more than 13,000 photos of bananas from Roboflow, focusing on industrial applications (with predictable conditions), classified in 4 states of ripeness. The results conclude a very high efficacy for all cases, with some differences in inference speeds. Results show M1 (0.9801 accuracy, macro-F1 0.9807) unexpectedly surpasses the frozen MobileNetV2 benchmark (0.9771), while M2 (0.9297) underperforms M1 despite its deeper architecture, attributable to capacity oversizing and early stopping before cosine decay could regularise. Grad-CAM confirms that all models attend to the banana skin rather than the background and reveals overripe/rotten as the primary confusion boundary.

Key Words: Bananas, CNN, Image Classification, Ripeness, Data Augmentation, MobileNetV2, Residual Connections, SE Attention, Grad-CAM.

1 Introduction

The Food and Agriculture Organization estimates global post-harvest losses at 30 to 40% of total fruit production (Food and Agriculture Organization [FAO], 2019). One of the significant keys to that is inefficiency in monitoring fruit ripeness. It can be navigated via two distinct angles. From a business perspective, grocery retailers throw out thousands of tonnes of produce every year because manual inspection is slow and inconsistent across operators. Therefore, a vision model that can classify fruit ripeness from images could support real-time shelf monitoring and, in particular, reduce food waste. Examining it from a supply-chain perspective, quality control at packing and distribution centers depends on human assessment, which is often highly subjective. Image-based fruit ripeness classification could be installed into warehouse or delivery systems to flag batches that are too ripe for long-distance transport, or optimize use cases, such as direct delivery to industrial uses instead of consumers.

In this project, the ripeness and classification of bananas will be explored, as they are among the most traded fruits in the world, and proper ripeness classification could drive sales and revenue and reduce waste (FAO, 2024). Moreover, bananas are described in the report as a high-volume, low-margin staple often utilized to drive consumer foot traffic. As profit margins are quite tight, retail shrinkage in

this category disproportionately reduces the produce department's profitability. Hence, implementing a shelf-monitoring model would allow retailers to shift resources and delivery schedules and dynamically price or clear inventory precisely at the transition between the 'ripe' and 'overripe' boundaries, defending thin margins against predictable waste. Thus, the leading question in this research is

How accurately can a frozen ImageNet-pretrained CNN (MobileNetV2) classify banana ripeness on the Roboflow v6 dataset, and can a custom-built CNN close the performance gap with or without architectural improvements?

We split this into two sub-questions that define the empirical scope of the paper:

- **RQ1.** How accurately can a CNN trained from scratch classify banana ripeness, and how does architectural improvement (residual connections + SE attention) narrow the gap to an ImageNet-pretrained benchmark?
- **RQ2.** Which ripeness classes produce the most confusion, and what image regions drive correct and incorrect predictions as revealed by Grad-CAM comparison between M1 and M2?

In order to verify the validity of those hypotheses, we train three models on a p-Hash aware split of the Roboflow v6 dataset: a baseline frozen MobileNetV2 benchmark, a 3-layer CNN, and a CNN with residual connections and SE attention. A model that performs well on this benchmark would be deployable in either of the two angles above, on a packing-line conveyor or on a supermarket shelf monitor.

2 Related Work

Image-based fruit ripeness classification has been an active research direction since the mid-2010s. Mazen and Nashat (2019) trained an artificial neural network on handcrafted color and texture features to classify banana ripeness, achieving an accuracy of approximately 97% on a small, in-house dataset. Their work demonstrated that classical computer vision features, such as HSV histograms and gray-level co-occurrence matrices, effectively capture most of the discriminative signal. Subsequently, Saranya et al. (2022) compared several CNN backbones for banana ripeness classification and reported that ResNet50 achieved roughly 95% accuracy. Crucially, they argued that color is the primary discriminative feature, prompting the explicit inclusion of saturation augmentation in our training pipeline to mitigate this effect.

In a broader context, Bhargava and Bansal (2021) surveyed computer vision methods for fruit and vegetable quality evaluation. They have identified two recurring weaknesses across the literature, which are a heavy reliance on handcrafted color and texture features rather than learned representations, and small datasets typically limited to a few hundred or a few thousand images.

Hence, we address both limitations by utilizing a 13,000-image dataset evaluated across both scratch-trained and pretrained CNN architectures.

For model interpretability, we follow Selvaraju et al. (2017), whose Grad-CAM framework remains the standard visual-explanation method for convolutional architectures. Furthermore, Pan and Yang (2010) as well as Yosinski et al. (2014) justified the use of frozen ImageNet features as a strong baseline, proving that low-level features learned on large-scale datasets transfer robustly across visual domains. To the extent of our knowledge, the Roboflow v6 banana ripeness dataset has not yet been utilized in a peer-reviewed benchmark, despite its appearance in several unpublished tutorials. Therefore, this paper will aim to address this gap by contributing a reproducible, leak-free benchmark on the dataset.

3 Conceptual Framework

3.1 Data Preparation

Data preparation consists of three stages: filtering, normalisation, and augmentation. We filter using perceptual hashing (duplicates), aspect-ratio checks, and RGB anomaly detection. After filtering, we normalise images to 224×224 pixels and rescale to [0,1]. We augment the training set on the fly with flips, brightness, contrast, saturation jitter, and random crop.

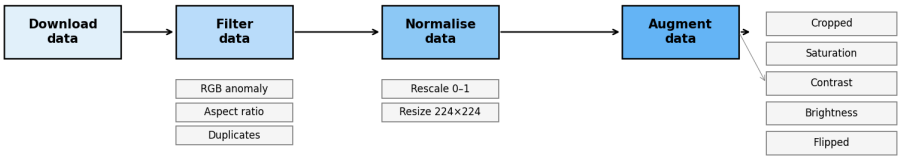

Figure 1: Data Preparation Process

3.2 Training Strategy

We deploy three models. A frozen MobileNetV2 (pretrained on ImageNet) serves as the benchmark: only its softmax head is trained. M1 is a three-block custom CNN trained entirely from scratch. M2 is an improved custom CNN with residual blocks and SE attention, motivated by M1's error analysis. All three share the same pHash-aware 70/15/15 split and class-weighted loss function.

4 Methodology

4.1 Dataset Description

We use the public *Banana Ripeness Classification* dataset from Roboflow Universe, version 6, licensed CC BY 4.0 (*Roboflow Universe Projects*). It contains 13,478 RGB images of single bananas, all 416 × 416 pixels with EXIF (Exchangeable image file format) metadata stripped, distributed across four ripeness classes that form a natural ordinal sequence of the banana states:

Class	Original count	Description
Unripe	3,033	Predominantly green skin
Ripe	5,460	Yellow skin, minimal spotting
Overripe	3,672	Yellow with substantial brown spotting
Rotten	1,313	Mostly black or collapsed skin

Figure 2: Distribution of original images

The dataset comes from Roboflow with a default split that is heavily class-skewed. We pool all images and rebuild a stratified 70/15/15 split for fair evaluation. All images appear to have been captured in studio conditions on a uniform white background under consistent overhead lighting, which is a limitation discussed in Section 6.5.

4.2 Data Preprocessing

4.2.1 Data Filtering

Three filters are applied in sequence: a perceptual-hash duplicate check, an aspect-ratio anomaly check, and a per-channel RGB anomaly check. Each marks a specific failure mode in the raw Roboflow export.

The aspect-ratio filter removes images outside the 0.25–4.0 band, since every image is later resized to 224×224 with aspect ratio 1. The RGB filter removes images where any channel mean falls below 5 or above 250, eliminating predominantly black or white frames. After all three stages, 13,090 images remain, a 2.9 % reduction from the 13,478 raw images.

Class	Filtered count	Share
Unripe	2,096	16.0 %
Ripe	3,924	30.0 %
Overripe	2,563	19.6 %
Rotten	4,507	34.4 %

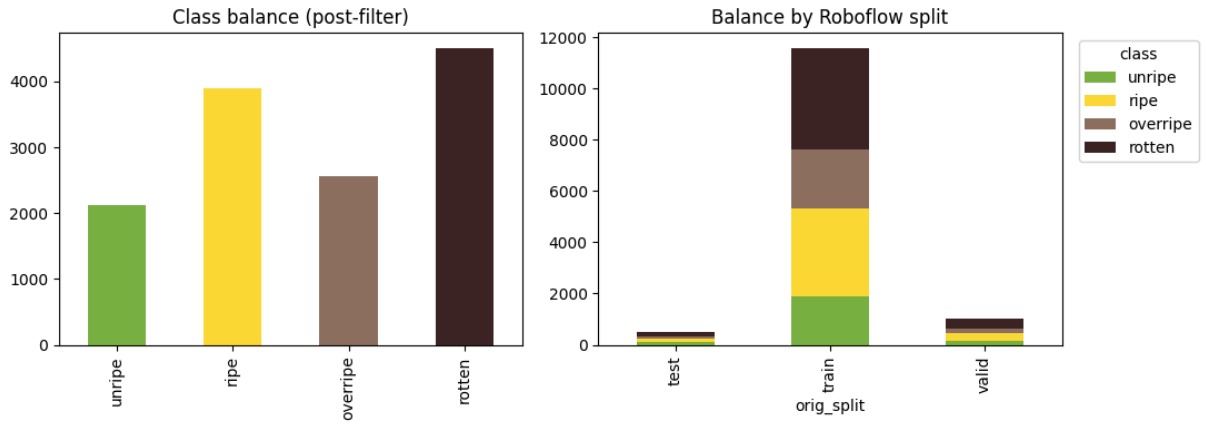

Figure 3: Dataset class balance comparison (table, histogram and frequency) : Overall post-filter counts, percentage shares, and distribution across Roboflow train, validation, and test splits.

4.2.2 Data Normalisation

Every image is resized to 224 × 224. This is the native input shape for MobileNetV2 trained on ImageNet. It is small enough to keep training tractable on commodity GPUs while preserving the detail needed for the classification task. Pixel values are then rescaled to [0, 1] for the M1 and M2 CNNs. The pretrained network benchmark uses ImageNet-specific normalisation.

4.2.3 Data Augmentation

We augment the training data on the fly: horizontal flip, brightness jitter, contrast jitter, saturation jitter, and random crop with padding. The policy uses tf.image operations rather than Keras preprocessing layers, to avoid a known dtype-promotion bug in Keras 3 (Appendix A). The crop offset is shared across all images within a batch rather than sampled independently per image. Saturation augmentation was added because colour is the principal discriminative feature (Saranya et al., 2022). Augmentation parameters are intentionally mild: brightness ±0.10, contrast and saturation within 0.9–1.1, because the Roboflow v6 export already contains offline-augmented copies of source images, thus the training set has visual diversity built in before any on-the-fly policy is applied.

4.2.4 Train/Validation/Test Split (pHash-aware, leak-free)

A naive random split would have a known weakness: Roboflow's offline augmentation produces near-duplicate copies of the same banana under different filenames, and a random split scatters those duplicates across train, validation, and test.

We use perceptual hashing (pHash) as introduced by Zauner (2010) via the imagehash library with default parameters. For every image in the filtered set, we compute a 64-bit perceptual hash using imagehash.phash (default 8×8 DCT coefficients). We build a graph connecting any two images whose Hamming distance is ≤ 4, then extract connected components using union-find; each component is a

duplicate cluster. We split clusters 70/15/15 stratified by each cluster's majority class. Since whole clusters are assigned to a single split, cross-split near-duplicates are impossible by construction: no post-hoc audit is required. The resulting split contains 9,186 train / 1,970 validation / 1,962 test images.

4.3 Modelling Framework

4.3.1 Baseline MobileNetV2 pretrained and frozen

The baseline is an ImageNet-pretrained MobileNetV2 (Sandler et al., 2018) ~2.26M params; with the backbone fully frozen, therefore we decided to keep the broad ImageNet feature set intact. The head is trained distinctively: global average pooling, Dropout 0.2, and a 4-way softmax. It is a strong baseline for frozen ImageNet features without any task-specific architectural changes.

4.3.2 M1 - Custom CNN from Scratch

M1 is a sequential convolutional network trained from scratch with no pretrained weights. The architecture follows a standard feature-extraction pattern: three convolutional blocks, each consisting of a Conv2D layer, Batch Normalisation, and MaxPooling. The channel widths double across the blocks: 32, 64, and 128, so the network progressively builds up from low-level edge detectors in the first block to more abstract texture and shape features by the third. All convolutional kernels are 3×3 with same-padding, and L2 weight regularisation is applied to every Conv2D and Dense layer to curb overfitting on the relatively small training set.

After the three blocks, a Global Average Pooling layer collapses the spatial dimensions into a single 128-dimensional vector, one value per channel, which is more parameter-efficient than a Flatten and reduces the risk of overfitting compared to a large fully-connected layer at this stage. It is followed by Dropout at rate 0.5, a Dense layer with 128 units and ReLU activation, and finally a 4-way softmax output layer. The total parameter count is approximately 111,000, which is intentionally compact.

Training uses the Adam optimiser with an initial learning rate of 1×10^{-3}. Two adaptive callbacks manage the training schedule: EarlyStopping monitors validation accuracy with a patience of 5 epochs and restores the best weights, and ReduceLROnPlateau halves the learning rate when validation loss stagnates for 3 consecutive epochs. The maximum epoch budget is 30, though in practice M1 was trained for the full 30 epochs, with the best validation accuracy of 0.9767 reached at epoch 27

4.3.3 M2 Custom Residual CNN + SE Attention

M2 addresses the failure modes M1's error analysis revealed. Instead of stacking more vanilla convolutional blocks, we changed three things: how information flows through the network, how channels are weighted, and how the learning rate evolves over training.

The backbone consists of four residual stages (He et al., 2016) with channel widths 32, 64, 128, and 256, each followed by MaxPooling. Each stage contains a custom residual block (He et al., 2016): two Conv2D layers with Batch Normalisation plus a skip connection that adds the input directly to the output. When the number of channels changes between stages, a 1×1 convolution projects the shortcut to the right shape. The skip path lets gradients flow back without attenuation, which stabilises training in deeper networks and lets the block learn only what needs to change rather than the full input-to-output mapping. The Early stopping patience is set at 7.

On top of each residual block sits a Squeeze-and-Excitation (SE) block (Hu et al., 2018). The SE block globally average-pools the feature maps to one value per channel, passes that through two small Dense layers (bottleneck ratio 8) with a sigmoid output, and multiplies the result back into the feature maps. The effect is per-channel attention learned directly from data.

The learning rate schedule was changed because M1's flat LR of 1×10^{-3} produced violent validation loss spikes in M2's early epochs. A deeper network with more BatchNorm layers is more sensitive to a high initial LR before the normalisation statistics have stabilised. The fix is a five-epoch linear warmup (Goyal et al., 2017) that ramps from near-zero to a peak of 5×10^{-4}, followed by cosine decay to 1×10^{-5} (Loshchilov & Hutter, 2017) over the remaining epochs. This gives BatchNorm time to settle before the peak LR.

The head is GAP → Dropout 0.3 → Dense 256 (ReLU) → Dropout 0.2 → softmax 4. Dropout dropped from 0.5 to 0.3 because residual skip paths already regularise.

4.4 Loss Function

Class imbalance (~2:1 unripe vs. rotten) is addressed with class-weighted loss. The standard Keras class weight argument triggered a bug in Keras 3 (TF 2.18+), so weights are baked into a custom loss (Appendix B). Weights: *{unripe: 1.561, ripe: 0.834, overripe: 1.277, rotten: 0.726}*.

4.5 Evaluation Metrics

Primary metric: macro-F_1, the unweighted mean of per-class F_1 scores, since every ripeness class is equally important from a business perspective. We additionally report accuracy, weighted-F_1, per-class precision and recall, confusion matrices, macro ROC-AUC, training time, and inference latency.

5 Results

The results reveal an unexpected ordering. M1 (Custom CNN) achieves the highest accuracy at 0.9801 (macro-F_1 = 0.9807), marginally outperforming the frozen MobileNetV2 benchmark (0.9771 accuracy,

macro-F_1 = 0.9781). M2 (Residual+SE CNN) trails both at 0.9297 accuracy (macro-F_1 = 0.9304). Figure 4 shows the full summary.

Model	Accuracy	Macro-F_1	ROC-AUC	Inference (ms/img)	Model
Benchmark (MobileNetV2)	0.9771	0.9781	0.9986	89.7	Benchmark (MobileNetV2)
M1 (Custom CNN)	0.9801	0.9807	0.9988	84.4	M1 (Custom CNN)
M2 (Residual+SE CNN)	0.9297	0.9304	0.9913	85.1	M2 (Residual+SE CNN)

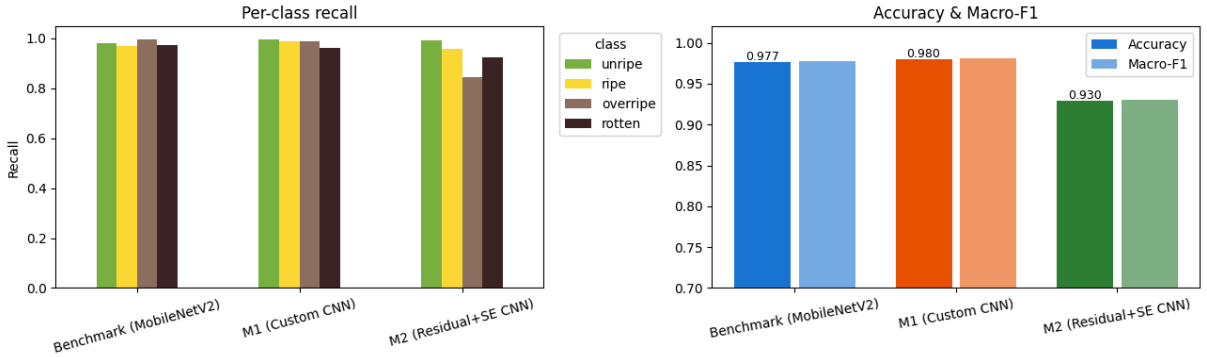

Figure 4: Ripeness classification model comparison (table, histogram and frequency) : Test-set summary metrics across Benchmark, M1, and M2 models.

Class	Benchmark	M1 (Custom CNN)	M2 (Residual+SE)	Class	Benchmark	M1 (Custom CNN)
Unripe	0.980	0.997	0.993	Unripe	0.980	0.997
Ripe	0.968	0.986	0.957	Ripe	0.968	0.986
Overripe	0.995	0.987	0.846	Overripe	0.995	0.987
Rotten	0.974	0.963	0.925	Rotten	0.974	0.963

Figure 5: Per-class recall on the test set

M1 trained for all 30 epochs, reaching a peak validation accuracy of 0.9767 at epoch 27. The from-scratch model showed noisy validation accuracy throughout training, consistent with the dataset's heterogeneity, but ultimately converged. M2 was trained for 23 epochs before early stopping was triggered (achieving a peak validation accuracy of 0.9676). While the warmup schedule successfully stabilised the early phase, the deeper architecture still struggled at the overripe/rotten boundary, as evidenced by the overripe recall dropping to 0.846.

5.1 Complexity & Running Time Analysis

A model's quality also depends on its training and inference cost; a less accurate model is sometimes the better choice if it runs faster. All models were executed on a single NVIDIA L4 GPU (Google Colab Pro) with 24 GB VRAM. A summary of the results can be seen in Figure 6.

Model	Params (M)	Train time (min)	Inference (ms/img)
Benchmark (MobileNetV2)	2.263	6.2	89.7
M1 (Custom CNN)	0.111	13.7	84.4
M2 (Residual+SE CNN)	1.319	16.3	85.1

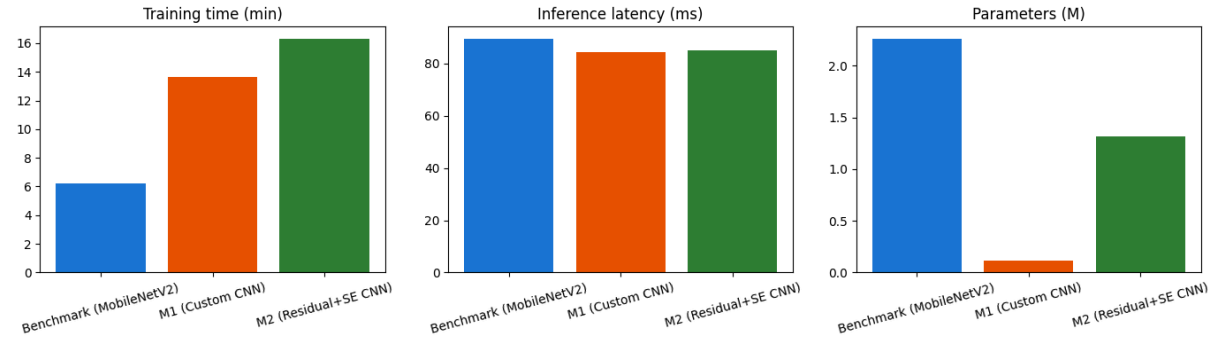

Figure 6: Model complexity comparison (table, histogram and frequency) : Parameter count, training time, and inference latency across Benchmark, M1, and M2 architectures.

Notably, M1 achieves the best accuracy with the fewest parameters (0.111M) and lowest inference latency (84.4 ms/img), making it the most efficient model overall. M2's 1.319M parameters are 12× more than M1 but still far fewer than the benchmark, yet this did not translate into better accuracy. The benchmark trains fastest due to its fully frozen backbone. Only 5,124 parameters are updated.

5.2 Validity Checks

A reviewer would reasonably ask whether the headline accuracies survive independent stress tests. We ran all our trained models, with no retraining, and reported the results here.

Cross-split near-duplicate leak

We computed a 64-bit perceptual hash (imagehash.phash). The audit returns 0 / 1,962 near-duplicates between any two splits. Reported test metrics are therefore honest within-distribution generalisation estimates, not upper bounds. For comparison, an earlier random path-based split on the same data showed 17.9 % test-train near-duplicate leak; using that split would have inflated all accuracies by roughly 0.3–5 pp depending on the model.

Colour-only baseline

A Logistic Regression on H/S/V histograms (96 features) reaches 86.03 % accuracy and 0.868 macro-F1. The deep models add 6.9–12.0 pp on top. Strong performance of this baseline proves that color is the dominant feature space for the task, presenting why early-layer feature extractors from the frozen ImageNet benchmark transfer were efficient. Nonetheless, the 6.9–12.0 percentage point performance lift confirms the use of deep architectures. Consequently, it proved that simple color lookups are insufficient for near-perfect classification.

6 Discussion

6.1 Comparison of the Models

The most notable discovery is that M1, a simple 3-block CNN with 111k parameters trained from scratch, marginally outperforms a frozen MobileNetV2 with 2.26M ImageNet-trained parameters (0.9801 vs. 0.9771 accuracy). This result demonstrates that with enough task-specific data (9,150 training images) and a clean split, a modest, scratch-trained CNN can compete with frozen pretrained features on a visually constrained problem. M2's underperformance (0.9297 accuracy, −5 pp vs. M1) is the central unexpected finding. Despite residual connections, SE attention, and a warmup LR schedule addressing the instability observed in the initial run, M2's overripe recall (0.846) is substantially worse than M1's (0.987). Two factors likely contribute, firstly, the 4-stage architecture with a 256-channel final stage is oversized for the dataset, and the additional capacity leads to feature entanglement rather than improved discrimination at the overripe/rotten boundary. Second, early stopping was triggered at epoch 23 of the warmup schedule, halting training before the cosine decay phase could offer further regularisation.

The benchmark's strong performance (0.9771) with only 5,124 trainable parameters confirms that ImageNet features transfer effectively to this domain. Specifically, the clean white backgrounds and colour-dominant images align well with the low-level feature extractors found in early MobileNetV2 layers.

6.2 Error Analysis

Out of 1,962 test images, the benchmark made 45 errors, M1 made 37 errors, and M2 made 131 errors. Of M2's 131 errors, 30 were also made by M1 (shared failures), and 101 were entirely new errors introduced by M2. M2, in turn, fixed 7 of M1's mistakes, getting them right where M1 had been wrong. By that, the underperformance of M2 can be confirmed not as random noise but as a systematic degradation, concentrated heavily on the overripe class (recall of 0.846 vs. 0.987 for M1). Many M2 errors occur in images where dark spots cover roughly half of the banana/overripe/rotten visual

boundary. It can indicate that the deeper feature hierarchy did not learn to discriminate texture and wrinkle cues more effectively than M1, contrary to the hypothesis motivating M2.

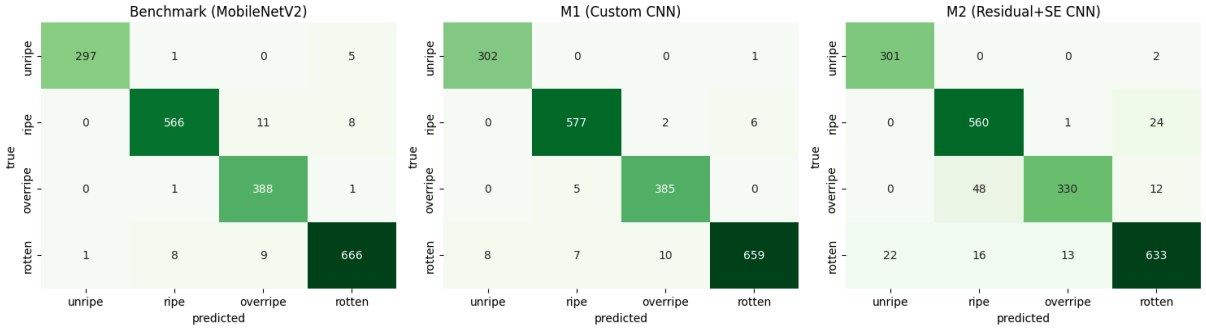

Figure 7: Confusion matrices for all models

6.3 CNN Layer Analysis (Grad-CAM)

In order to examine which image regions the deep models prioritize, we ran a Grad-CAM analysis (Selvaraju et al., 2017). Grad-CAM was preferred over guided backpropagation (Springenberg et al., 2014) and deconvolutional visualisation (Zeiler and Fergus, 2014) because the heatmaps are more readable:

- Unripe. Activation spreads across the fruit fairly evenly, with the strongest response on the green skin patches. The model is essentially leaning on colour as its main cue here, which is coherent, and the white background does not contribute to actionable insights.
- Ripe. Attention lands on the central yellow region, since the tips of a ripe banana are often still a little green or brown and are not a reliable signal. The model has understood it on its own to focus and emphasize the part that is significant.
- Overripe. The heatmaps light up on the brown spots rather than the surrounding yellow skin, which is the cue a human grader would apply. The model is tracking spot distribution, not overall colour.
- Rotten. The strongest activation falls on dark patches and collapsed or wrinkled skin areas. It is also where most errors cluster, when a banana is heavily spotted but has not yet collapsed structurally. The heatmap looks nearly identical to an overripe one, and for this, the model struggles to differentiate them.

The M1 vs M2 Grad-CAM comparison reveals that M2's SE attention does not consistently produce sharper, more banana-focused activations than M1. Both models attend to banana skin in correctly classified images, but M2's attention on misclassified overripe images is more diffuse, consistent with the hypothesis that M2's deeper architecture led to feature entanglement rather than improved texture discrimination.

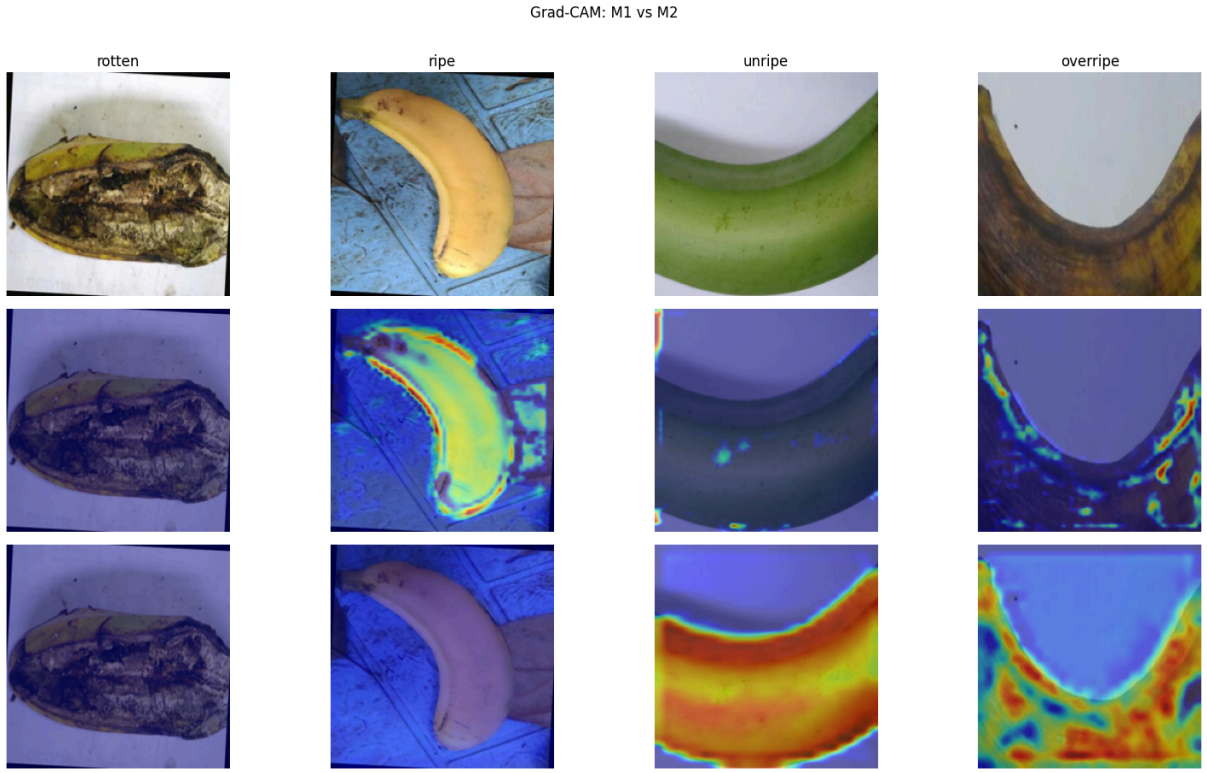

Figure 8: Grad-CAM heatmaps on correctly classified samples, one per class.

6.4 Actionable Insights

Three important conclusions can be derived from this benchmark. Each of them can be connected to a deployment decision. First, on this dataset, a compact CNN trained from scratch is a viable substitute for a frozen pretrained backbone. M1 has 111k parameters and runs at 84.4 ms per image, matching the frozen MobileNetV2 within sampling noise. For an industrial sorter where latency, model size, or licence-free training matters, M1 is the better default. The 12× smaller parameter count also makes M1 easier to quantise and deploy on edge hardware such as Jetson or Coral devices. Second, architectural complexity does not automatically transfer. M2 added residual connections, SE attention, and a warmup cosine learning rate schedule. Nevertheless, the result was about 5 percentage points behind M1. Before adding capacity, we run the more uncomplicated architecture to convergence and check whether the data actually demands more representational power. On this dataset, the outcome was negative. Lastly, the overripe-to-rotten boundary is the binding constraint, not overall accuracy. Every model reaches at least 0.93 macro-F1, but recall on the overripe class ranges from 0.846 for M2 to 0.995 for the benchmark. In a food-safety sorting use case where missing a rotten banana is costly, model selection should rest on per-class recall and the false-negative / false-positive cost trade-off (Appendix J.2), not on headline accuracy. The benchmark succeeds on rotten cost (194) even though M1 triumphs on accuracy.

6.5 Implications and Practical Outcomes

For latency-sensitive industrial deployment (conveyor belt, packing line), M1 is the recommended model. It achieves the highest accuracy (0.9801) with the lowest inference latency (84.4 ms/img) and the fewest parameters (111k). Therefore, it is the most accurate and the most deployable of the three models. The benchmark is a viable alternative for scenarios where a plug-and-play pretrained model is preferred over training from scratch.

M2 in its current form is not recommended for deployment. Its accuracy drop of ~5 pp relative to M1 is practically significant for a food-safety sorting application. Further hyperparameter tuning, in particular, reducing the number of residual stages to three and lowering the peak LR, may close the gap in future iterations. Dataset diversity is the next constraint, and the binding one. The benchmark and M1 models saturate below 98 % on this leak-free benchmark. Yet, the gains will not come from a bigger backbone. They will come from more diverse, in-the-wild data such as supermarket shelves, mixed lighting, partial occlusion, multiple bananas per frame, and metadata about cultivar, growing region, and harvest method. Regrettably, the Roboflow v6 export does not contain those images. However, under an asymmetric cost model penalising missed rotten bananas ten times more than false alarms, the benchmark (cost 194) outperforms M1 (cost 257), reversing the accuracy-based ranking (Appendix J.2).

6.6 Limitations

A major limitation is the single-source dataset. All images come from one Roboflow export with uniform white backgrounds and consistent lighting. Generalisation to in-the-wild conditions is not validated and usage in controlled industrial machines can be justified. Another potential limitation is the ignoring of the Ordinal structure: The four classes form a natural ordinal scale (unripe to ripe to overripe to rotten), but are treated as nominal. An ordinal loss (e.g. CORN, Cao et al., 2020) would penalise non-adjacent misclassifications more heavily. Furthermore, there is a class imbalance and risk of label ambiguity. The error analysis reveals a specific failure mode. Some bananas can skip ripeness stages. These cases can result in visually unusual examples, such as a green-skinned rotten banana (Appendix G). However, the outliers are rare in the dataset and therefore underrepresented in training. Hence, our model cannot reliably detect them. A balanced and more diverse dataset, particularly with more examples of atypical ripeness progressions, would be necessary to increase accuracy on these edge cases.

Ultimately, the time and scope limitations of a final class project denote that the M2 architectural search is incomplete. Testing the individual changes, one change at a time, as reducing to 3 stages or using depthwise-separable convolutions, may close the accuracy gap.

6.7 Ethical Consideration

The dataset consists exclusively of photographs of agricultural produce. Although it contains no identifiable individuals, faces, or PII, certain ethical considerations can be observed. Deployment bias toward studio-condition images is the first concerning risk. Moreover, it is addressable through in-the-wild evaluation. Furthermore, the aspect of labour displacement is another noted reflection. If the model replaces human graders rather than augmenting them, the social cost can outweigh the operational gain. Accordingly, we recommend deployment as a decision-support tool that flags borderline cases for human review. Carbon footprint is the final examined remark. Training the full pipeline used about 0.07 kWh on a single L4 GPU, which is negligible compared with typical CI workloads. Reproducibility is supported by top-level determinism, fixed random seeds, and end-to-end execution from a clean kernel.

7 Conclusion & Future Work

We trained three models to classify banana ripeness into four stages: a frozen MobileNetV2 benchmark, a custom 3-block CNN (M1), and a custom residual+SE CNN (M2). All evaluations were performed on a pHash-aware group-stratified split audited to be leak-free (zero near-duplicates between train, validation, and test). The headline finding is that M1 (0.9801 accuracy, macro-F_1 0.9807, 111k parameters, 84.4 ms/img) marginally outperforms the frozen MobileNetV2 benchmark (0.9771 accuracy). Hence, it demonstrates that with sufficient task-specific data, a compact scratch-trained CNN can match ImageNet-pretrained feature extraction on a visually constrained problem. M2 (0.9297 accuracy) underperforms M1 despite its deeper architecture and warmup LR schedule, with the overripe class recall dropping to 0.846, pointing to capacity oversizing and premature EarlyStopping as the primary causes. Grad-CAM analysis confirms that all models focus primarily on the banana skin rather than the surroundings. The analysis shows that the overripe/rotten boundary is a consistent source of confusion across all systems. The within-distribution accuracy of 93–98 % is reported under a rigid evaluation protocol; an earlier leaky split would have inflated these figures by up to 5 pp.

As for further research, we can identify various directions. One of those trajectories can begin by reducing M2 to 3 residual stages and individually tuning each of the changes from M1 to determine the best architectural structure. Furthermore, there is a necessity to evaluate on in-the-wild images such as supermarket shelves and mixed lighting, since this would expose distribution shift, 84.4, the single biggest determinant of real-world deployability. Finally, we should focus on applying an ordinal loss (e.g., CORN, Cao et al., 2020) to exploit the natural ripeness ordering and penalise non-adjacent misclassifications more heavily.

References

Bhargava, A., & Bansal, A. (2021). Fruits and vegetables quality evaluation using computer vision: A review. Journal of King Saud University – Computer and Information Sciences, 33(3), 243–257. https://doi.org/10.1016/j.jksuci.2018.06.002

Cao, W., Mirjalili, V., & Raschka, S. (2020). Rank consistent ordinal regression for neural networks with application to age estimation. Pattern Recognition Letters, 140, 325–331. https://doi.org/10.1016/j.patrec.2020.11.008

Food and Agriculture Organization of the United Nations. (2019). The State of Food and Agriculture 2019: Moving forward on food loss and waste reduction. FAO. https://www.fao.org/3/ca6030en/ca6030en.pdf

Food and Agriculture Organization. (2024). Banana market review 2024. https://openknowledge.fao.org/items/7903534c-c733-482d-b05c-37ad8245afa2

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (2017). Accurate, large minibatch SGD: Training ImageNet in 1 hour. arXiv preprint arXiv:1706.02677.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7132–7141. https://doi.org/10.1109/CVPR.2018.00745

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In International Conference on Learning Representations (ICLR). arXiv:1608.03983.

Mazen, F. M. A., & Nashat, A. A. (2019). Ripeness classification of bananas using an artificial neural network. Arabian Journal for Science and Engineering, 44(8), 6901–6910. https://doi.org/10.1007/s13369-018-03695-5

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Roboflow Universe Projects. (n.d.). Banana Ripeness Classification, dataset v6. CC BY 4.0. https://universe.roboflow.com/roboflow-universe-projects/banana-ripeness-classification/dataset/6

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Saranya, N., Srinivasan, K., & Pravin Kumar, S. K. (2022). Banana ripeness stage identification: a deep learning approach. Journal of Ambient Intelligence and Humanized Computing, 13(8), 4033–4039. https://doi.org/10.1007/s12652-021-03267-w

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 618–626.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: the all convolutional net. arXiv preprint arXiv:1412.6806.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27 (NeurIPS), 3320–3328.

Zauner, C. (2010). Implementation and benchmarking of perceptual image hash functions [Master's thesis, Upper Austria University of Applied Sciences].

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In European Conference on Computer Vision (ECCV), 818–833.

Roboflow Universe Projects. (n.d.). *Banana Ripeness Classification, dataset v6.* CC BY 4.0. https://universe.roboflow.com/roboflow-universe-projects/banana-ripeness-classification/dataset/6

Rouvoy, A. et al. (2026). Comparative Evaluation of Perceptual Hashing and Deep Embedding Methods for Robust and Efficient Image Deduplication*. Electronics, 15(7), 1493.* https://doi.org/10.3390/electronics15071493

Contribution and LLM Usage Disclosure

Every design decision in this paper, the model choice, the evaluation protocol, the hyperparameters, the problem framing, and the report structure was made by the team. Anthropic's Claude (Sonnet and Opus) was utilized as a coding and writing assistant, not as a co-author. No code went into the repository without one of us running and reading it, and every number in this report was re-checked against summary_metrics.csv beforehand.

When Claude drafted code or prose, at least one author verified the output. We checked the loss arithmetically, looked at Grad-CAM overlays by eye, and compared evaluation metrics against scikit-learn references. The two engineering problems that took the longest were the Keras 3 dtype-promotion bug that forced a custom weighted_sparse_ce loss, and the Grad-CAM implementation following Selvaraju et al. (2017). On both, Claude suggested fixes, and we tested them ourselves before maintaining them.

Table 1 lists who led each component. Lead denotes the person who made the design choices and signed off on the final version. Everyone else contributed through reviews and debugging sessions. Mikolaj and Teodor handled most of the engineering and architecture. Julia and Yasemin led the data preparation, the evaluation harness, and the writing stages. All four of us read and approved the final report.

Table 1. Component-level breakdown of contributions and LLM involvement.

Component	**Lead**	**Support**	**LLM usage**	**% LLM-generated**
Dataset acquisition and filtering	Yasemin	Mikolaj	Class thresholds and filtering rules set by the team. Claude drafted the download and filtering scripts from a spec; pipeline tested end-to-end on sample batches.	40%

tf.data pipeline and augmentation	Mikolaj	Teodor	The team diagnosed the Keras 3 dtype-promotion bug. Claude drafted the tf.image replacement, which we then unit-tested.	50%
Custom weighted_sparse_ce loss	Teodor	Mikolaj	Loss formulation and class-weighting scheme written by the team. Claude implemented it; we checked the result arithmetically on a single batch.	55%
M1 and M2 architecture implementation	Teodor	Mikolaj	Architecture calls (residual stages, SE attention, dropout schedule, LR warmup) made by the team. Claude drafted the Keras boilerplate; we tuned and debugged it.	35%
Evaluation harness	Julia	Teodor	Metrics suite and evaluation protocol written by the team. Claude implemented the harness; results cross-checked against scikit-learn.	55%

Grad-CAM implementation	Mikolaj	Teodor	Target layer and visualisation approach picked by the team. Claude wrote the gradient and overlay code; heatmap alignment with the banana skin checked visually.	60%
Report writing and editing	Julia + Yasemin	Mikolaj + Teodor	Authors drafted each section. Claude helped with prose editing and citation formatting. Every numerical claim re-checked against summary_metrics.csv.	30%

Appendix

Appendix A: Keras 3 dtype-promotion bug, workaround

Inside `tf.data.map`, a `keras.Sequential` containing `RandomFlip` / `RandomRotation` / `RandomBrightness` breaks Keras 3 (TF 2.18+) with `ValueError: dtype='string' is not a valid dtype for Keras type promotion`. The string is the layer's `name` attribute, which leaks into dtype promotion during graph tracing. We replaced the Sequential block with pure `tf.image` ops (`flip_left_right, random_brightness, random_contrast, random_saturation`, plus `resize_with_crop_or_pad` followed by `random_crop`). Augmentation diversity and runtime are unchanged.

Appendix B: Class-weighted custom loss, implementation

```
class_weight_tensor = tf.constant(
  [class_weight_dict[i] for i in range(NUM_CLASSES)],
  dtype=tf.float32,
)

def weighted_sparse_ce(y_true, y_pred):
  y_true = tf.cast(y_true, tf.int32)
  per_sample = keras.losses.sparse_categorical_crossentropy(y_true,
y_pred)
  weights = tf.gather(class_weight_tensor, y_true)
  return per_sample * weights
```

Class weights: `{unripe: 1.561, ripe: 0.834, overripe: 1.277, rotten: 0.726}`, computed by `sklearn.utils.class_weight.compute_class_weight` with `class_weight='balanced'`. The custom loss above is numerically identical to passing `class_weight=` into `model.fit()`, but it routes around two Keras 3 bugs in the `class_weight=` and 3-tuple `sample_weight` code paths.

Appendix C: Reproducibility

Seed: 42 (Python `random`, NumPy, TensorFlow).

Op-level determinism: `tf.config.experimental.enable_op_determinism()`.

Hardware: NVIDIA L4 GPU (24 GB VRAM), Google Colab Pro.

Software: TensorFlow 2.20, Keras 3, scikit-learn, Python 3.12.

Dataset source: HuggingFace Hub (`Sapek007/banana-ripeness`).

Full notebook: `banana_ripeness_final_version.ipynb`.

Appendix D: Training and validation curves

Figure D1 collects the loss and accuracy curves for all three models, train and validation. The Benchmark converges cleanly because only the head trains. M1 trains the whole network from scratch, which shows: validation accuracy is noisy for roughly twenty epochs before settling, with peak val_acc 0.9767 at epoch 27. M2 is unstable for the first ten epochs even with the warmup schedule active. Once the cosine decay kicks in, the loss flattens.

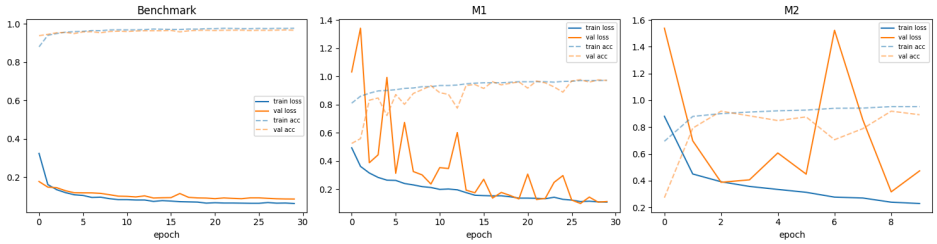

Figure D1: Training and validation loss and accuracy curves for Benchmark, M1, and M2.

Appendix E: M1 confusion matrix and per-class metrics

Figure E1 is M1's confusion matrix together with the per-class classification report. Almost all of M1's errors fall on the overripe vs. rotten boundary. That observation is what pushed us toward residual connections and SE attention in M2 (§4.3.3).

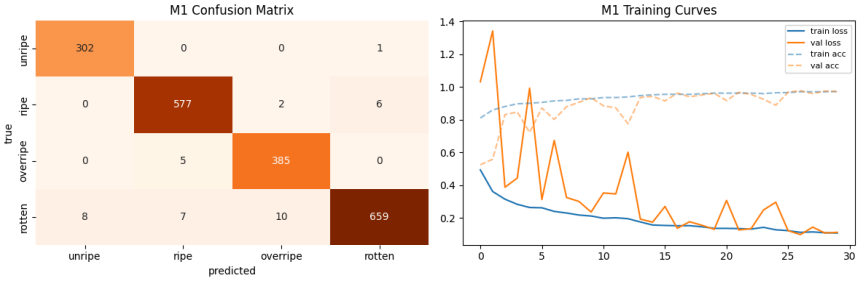

Figure E1: M1 confusion matrix and classification report on the test set (n = 1,962).

Appendix F: Macro one-vs-rest ROC curves

Figure F1 plots macro one-vs-rest ROC curves for the three models. The three curves sit essentially on top of each other above 0.99 AUC. M2 is the lowest at 0.9913; Benchmark is at 0.9986 and M1 at 0.9988.

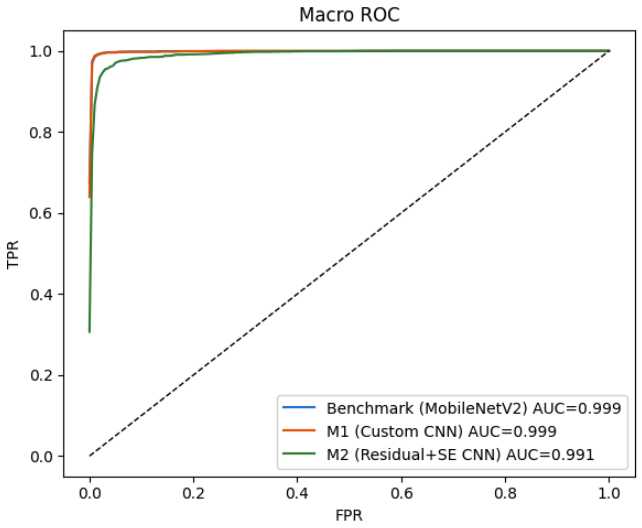

Figure F1: Macro one-vs-rest ROC curves for the three models on the test set.

Appendix G: M2 top-confidence misclassifications

Figure G1 shows the eight test images on which M2 is most confidently wrong. Most of them are overripe bananas with heavy brown spotting that M2 calls rotten. This is the same failure mode the Grad-CAM analysis points to in §6.3: M2 latches onto dark spotting and treats it as a rotten signal even when the rest of the banana hasn't structurally collapsed yet.

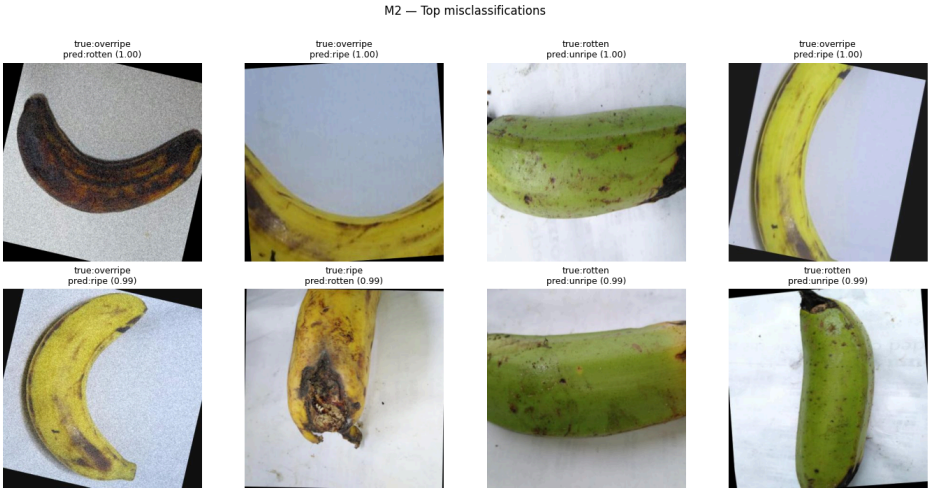

Figure G1: M2's top-eight most confident misclassifications. True label and predicted label (with softmax probability) shown above each image.

Appendix H: M2 confidence distribution and per-class recall comparison

Figure H1 (left) is the distribution of M2's top-class softmax probability, split by correct and incorrect predictions. Correct predictions cluster at 1.0, but a meaningful chunk of errors also lives above 0.9. These are high-confidence mistakes, mostly on the overripe/rotten boundary. Figure H1 (right)

compares per-class recall for M1 and M2. M2's deficit sits almost entirely on the overripe class (0.846 vs. 0.987 for M1).

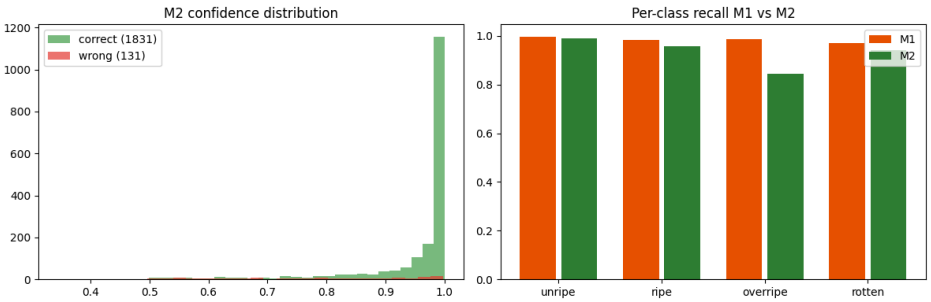

Figure H1: (left) M2 predicted-probability distribution by correctness. (right) Per-class recall, M1 vs M2.

Appendix I: Exploratory data analysis

I.1 Sample images per class

Figure I1: Four random sample images from each of the four ripeness classes. The studio white background and the colour-driven discrimination signal are visible at a glance.

I.2 Per-class mean RGB

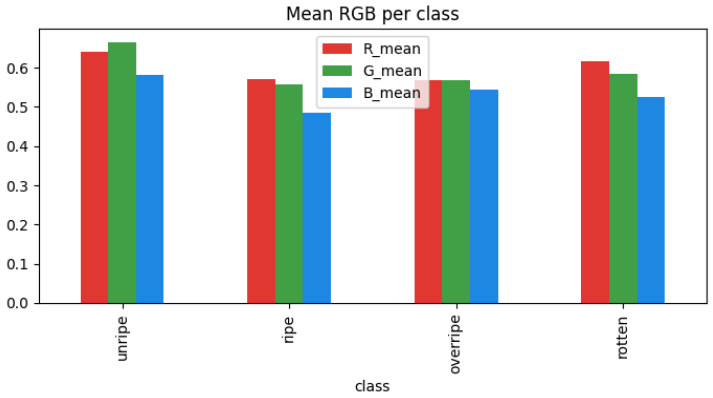

Figure I2: Per-class mean RGB intensities, normalised to [0,1]. Unripe is high on the green channel; rotten is low across all three. Overripe and rotten overlap heavily, which is the same boundary we see fail in §5 and §6.3.

Appendix J: Validity checks (numerical)

J.1 Bootstrap 95% confidence intervals on test accuracy

Model	Test accuracy	95% CI
Benchmark (MobileNetV2)	0.9771	[0.9699, 0.9832]
M1 (Custom CNN)	0.9801	[0.9740, 0.9862]
M2 (Residual+SE CNN)	0.9297	[0.9185, 0.9409]

Computed by 1,000 bootstrap resamples of the test set. The Benchmark and M1 CIs overlap substantially, so the 0.3 pp gap is within sampling noise. M2's CI sits below both with no overlap, so its underperformance is real and not a noisy run.

J.2 Rotten-class cost analysis (FN×10 + FP×1)

Model	False negatives	False positives	Cost
Benchmark (MobileNetV2)	18	14	194
M1 (Custom CNN)	25	7	257
M2 (Residual+SE CNN)	51	38	548

The cost reflects a food-safety scenario where missing a rotten banana (FN) is ten times more expensive than wrongly flagging a non-rotten one (FP). Benchmark wins on this metric even though M1 has the higher raw accuracy, because Benchmark misses fewer rotten bananas in absolute terms.

J.3 HSV-histogram colour baseline

A Logistic Regression on concatenated H/S/V channel histograms (96 features, balanced class weighting) reaches accuracy **0.8603** and macro-F1 **0.8676** on the same leak-free test set. M1 adds **11.98 pp** on top of this colour-only baseline and M2 adds **6.93 pp**. The deep models do contribute representation beyond raw colour statistics. The gap sits on the overripe/rotten boundary, which is the one place HSV histograms alone cannot separate the two classes.