SIN-SMFL: AN ABLATION STUDY OF MODULATION, CLASS BALANCING, AND GAIN CALIBRATION IN DETECTION LOSSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Modulated classification losses (e.g., Focal Loss) and class-frequency reweighting schemes have each been studied for object detection, but their *composition* and the practical hyperparameter interactions it introduces remain underexplored. We conducted a study of sine-modulated classification loss combined with effective-number class balancing (Sin-SMFL) on YOLOv8 using the COCO 2017 object detection dataset, comprising 27.6 GB of data with 118K training images, 5K validation images.We compared the six configurations across different classes using three random seeds with 15-epoch training to understand the effect of each design choice. We find that, first, combining sine modulation with class balancing underperforms the detector's own unmodified default loss. This gap occurs because sine modulation reduces the magnitude of the classification loss compared with binary cross-entropy (BCE). As a result, a gain tuned for BCE gives too little weight to the modulated loss. Second , once this gain is corrected, sine modulation is a robust improvement over classical power-law (Focal Loss) modulation, improving 77 of 80 COCO classes when balancing and gain are held fixed. Third, a learned loss balancer automatically fixes most of the loss-weighting problem without hyperparameter search, but manual tuning still performs better for some individual classes. Fourth, effective-number class balancing mainly improves performance for the 10–15 rarest classes, but it does not reduce the overall performance gap between different classes.We report these findings, including the negative and mixed results, as a case study showing that loss-function modifications do not always combine effectively under realistic, limited training settings.

1 INTRODUCTION

Modulated classification losses, such as Focal Loss (Lin et al., 2017), and frequency-aware class reweighting schemes, such as effective-number-of-samples weighting (Cui et al., 2019), are commonly used to address class imbalance in object detection. A next step is to combine the two approaches: use a modulation function to give less importance to easy examples while also giving more weight to classes with fewer samples, all within the same classification loss.

We introduce Sin-SMFL, a sine-modulated, effective-number-balanced classification loss, and integrate it into YOLOv8's (Jocher et al., 2023) detection loss. Rather than reporting a single comparison against a baseline, we construct a six-configuration ablation grid that separately varies the modulation function (none, classical power-law/Focal, or the proposed sine form), the class-balancing scheme (uniform or effective-number), and the resulting loss terms are combined with the detector's box and distribution-focal-loss (a fixed manually-tuned gain, or a learned adaptive alternative). We evaluate every configuration with three seeds at a 15-epoch training budget.

Two questions motivate this design. First, does composing modulation and class balancing actually help, or do individually components interact in ways that cancel or reverse each other's benefit? Second, If the combined method performs worse than the default loss, is the problem with the modulation function itself, or is it because the classification-loss gain was tuned for the original loss and does not work well with the new loss?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

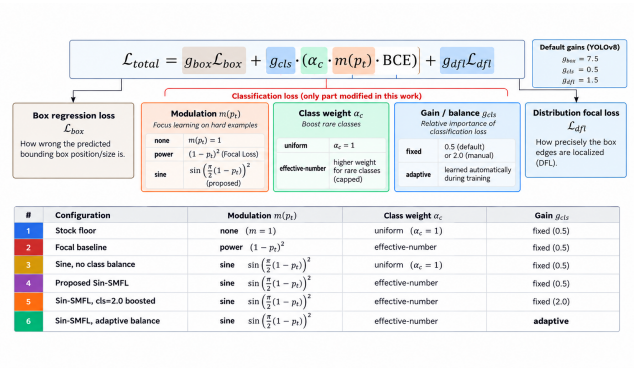

Figure 1: Study design: three customizable loss-function axes, six configurations, and the isolated pairwise comparisons used throughout Section 5.

We find that the answer to the first question is clearly no, as under 15-epoch training, our proposed configuration underperforms the unmodified default loss overall. But when we tested each design choice separately (Section 5.2), we found that the main problem comes from the second mechanism, not from the modulation function itself.After correcting the classification gain for the smaller loss magnitude, sine modulation performs consistently better than classical Focal Loss (Section 5.3).

This shows that the poor overall performance of the naive combination hides a real improvement from sine modulation at the class level. The motivation for sine modulation (Section 3.2) is that it keeps the gradient magnitude bounded, unlike classical power-law modulation at low γ. The comparison in Section 5.3 directly tests whether this idea improves performance. We also tested whether the gain correction can be learned rather than hand-tuned, via an uncertainty-weighted loss balancer (Section 5.5), and examine class balancing's effect at the level of individual classes rather than only in aggregate (Section 5.6).

Figure 1 summarizes the study design: the three customizable loss-function axes and the six resulting configurations.

2 Related Work

Focal Loss (Lin et al., 2017) introduced the modulation approach that we build on. It reduces the weight of each classification loss based on the model's confidence, using $(1 - p_t)^\gamma$. This prevents the large number of easy negative examples in object detection from dominating the training gradient. Our power-modulation baseline (Section 3.2) is this exact formulation, integrated into YOLOv8 rather than RetinaNet. Sine modulation (Section 3.2) is proposed as an alternative modulating function within this same framework, motivated by a bounded-gradient-magnitude argument rather than by a different detector or a different underlying loss.

Another approach deals with class imbalance directly, without modulation. Inverse-frequency weighting is a first approach but is unbounded as a class's sample count approaches zero and treats every additional instance of a class as equally informative. (Cui et al., 2019) address class imbalance using effective-number-of-samples weighting, which we use throughout this study (Section 3.3). The method gives less additional importance to repeated samples, while still giving higher weights to rare classes and keeping the weights bounded. Other approaches to the same underlying problem operate at different points in the pipeline. For example, margin-based losses widen decision boundaries for rare classes (Cao et al., 2019), while equalization losses suppress gradient updates from head classes on channels relevant to tail classes (Tan et al., 2020). Another approach decouples representation learning from classifier re-balancing entirely (Kang et al., 2020). These methods differ from our approach, which directly reweights the per-class terms of a fixed loss function. The class imbalance addressed by these methods is a first-order concern in large-vocabulary detection and recognition benchmarks (Gupta et al., 2019; Van Horn et al., 2018; Wang et al., 2017). We focus on effective-number weighting because it can be directly combined with a modulation function with-

out changing the detector architecture, making it well suited for studying the composition question addressed in this paper.

A detection loss has three parts: box regression, distribution focal loss, and classification. How we combine these parts is different from how we modify the classification loss itself. Fixed, hand-tuned gains are the default in most detector codebases, including YOLOv8's own g_{box}=7.5, g_{cls}=0.5, g_{dfl}=1.5. Kendall et al. (2018) learn weights for different tasks using homoscedastic uncertainty. Their method was originally designed to combine semantic segmentation, instance segmentation, and depth-regression losses in a single network. We adapt this formulation (Section 3.4) to combine the three loss terms of our detector. This provides a hyperparameter-free alternative to manually tuning g_{cls} when modulation changes the magnitude of the classification loss (Section 5.5). Chen et al. (2018) balance tasks by adjusting their weights based on the gradient magnitude of each task at a chosen network layer. Unlike our approach, which balances the loss terms themselves, GradNorm operates directly on gradients. We use the uncertainty-weighting formulation instead of GradNorm because it does not require choosing a shared network layer for gradient normalization. It also works as a drop-in replacement for the three scalar gains already used in YOLOv8's loss.

3 METHOD

3.1 LOSS FORMULATION

We modify only the classification term of YOLOv8's detection loss , box regression (CIoU) and distribution focal loss (DFL) are unchanged. For predicted logits x and the soft, IoU-scaled targets $y \in [0, 1]$ produced by YOLOv8's task-aligned assigner (both of shape $[\cdot, C]$ for $C = 80$ classes), we first generalize the standard focal-loss target-probability p_t to soft targets:

$$p = \sigma(x), \qquad p_t = p \cdot y + (1 - p) \cdot (1 - y) \tag{1}$$

which reduces to the textbook $p_t = p$ if y=1 else $1-p$ for hard binary labels, and varies continuously for YOLOv8's soft assigned targets. The classification loss is then an elementwise-weighted binary cross-entropy:

$$\mathcal{L}_{cls} = \frac{1}{\sum y} \sum_{i,c} \alpha_c \cdot m(p_{t,i,c}) \cdot \mathrm{BCE}(x_{i,c}, y_{i,c}) \tag{2}$$

where i ranges over anchors assigned by the task-aligned assigner, c over classes, α_c is a per-class weight (Section 3.3), $m(\cdot)$ a modulation function (Section 3.2), and the normalizer $\sum y$ is the assigner's target-score sum, unchanged from stock YOLOv8. Setting $m \equiv 1$ and $\alpha_c \equiv 1$ reduces $\mathcal{L}_{cls}$ exactly to plain `BCEWithLogitsLoss`, which is the stock-floor configuration in our ablation grid.

3.2 MODULATION FUNCTIONS

A modulation function $m : [0, 1] \to [0, 1]$ down-weights the loss contributed by easy, confidently-correct examples (p_t near 1) while leaving hard examples (p_t near 0) unmodulated. We require $m(0)$=1 and $m(1)$=0 for every variant.

No modulation (stock floor): $m(p_t) = 1$.

Power modulation (classical Focal Loss; Lin et al., 2017): $m(p_t) = (1-p_t)^{\gamma}$, with γ=2 throughout this study.

Sine modulation (proposed): $m(p_t) = \sin\big(\frac{\pi}{2}(1 - p_t)\big)^n$, with n=2 throughout this study.

Writing $x = 1 - p_t$ for "difficulty," a small-angle expansion gives $\sin(\frac{\pi}{2}x)^n \approx (\frac{\pi}{2})^n x^n$ as $x \to 0$: sine modulation agrees with power modulation up to a constant rescaling on easy examples, and the two families agree exactly at the hard boundary (x=1). They diverge in the interior of $[0, 1]$: at γ=n=2 (used throughout this study), sine's derivative exceeds power's for $x \lesssim 0.68$ and falls below it thereafter (Figure 2b) sine redistributes gradient magnitude toward mid-difficulty examples and suppresses it more aggressively near the hardest examples, rather than being uniformly smaller everywhere. More generally, for $\gamma < 1$ the derivative of x^{γ} is unbounded as $x \to 0^+$, whereas $\left|\frac{d}{dx}\sin(\frac{\pi}{2}x)^n\right| \le \frac{n\pi}{2}$ uniformly on $[0, 1]$ for every $n \ge 1$; this boundedness guarantee, rather than a claim of smaller gradients everywhere. Section 5.3 tests the resulting modulation shape experimentally by isolating modulation type from every other design choice.

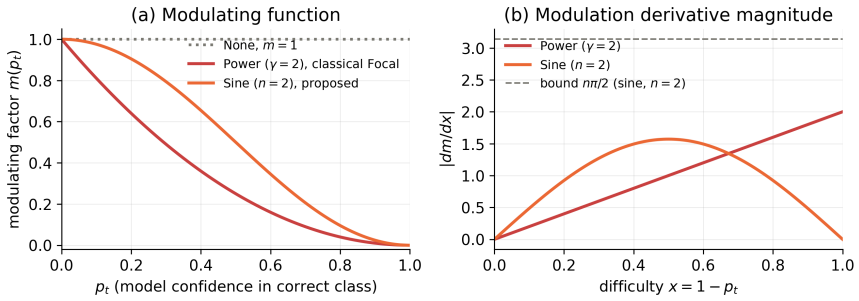

Figure 2: Sine vs. power modulation of the classification loss. (a) Modulating factor $m(p_t)$ for none, power (γ=2), and sine (n=2). (b) Derivative magnitude $|dm/dx|$ as a function of difficulty $x = 1 - p_t$; sine exceeds power for $x \lesssim 0.68$ and falls below it thereafter, both bounded by $n\pi/2$.

3.3 Class weighting

A class weighting scheme maps raw per-class training-set instance counts n_c to loss weights α_c.

Uniform (ablation floor): $\alpha_c = 1$ for all c.

Effective number of samples (Cui et al., 2019), the scheme used throughout this study wherever balancing is enabled: with $E_c = \frac{1-\beta^{n_c}}{1-\beta}$,

$$\alpha_c = \frac{1}{E_c} \Big/ \overline{\left(\frac{1}{E_c}\right)} \tag{3}$$

i.e. inverse effective number, renormalized to mean 1 across classes so the overall loss scale is unaffected by enabling balancing. We use β=0.999 throughout. As $n_c \to 0$, $E_c \to 1$ and α_c approaches a finite maximum rather than diverging, which distinguishes this scheme from naive inverse-frequency weighting.

Per-class training instance counts n_c are computed directly from the COCO2017 (Lin et al., 2014) train annotation file (min/median/max across the 80 classes: 198 / 6,069 / 262,465), not estimated or assumed.

3.4 Combining loss terms

The three per-batch loss terms — $\mathcal{L}_{box}$, $\mathcal{L}_{cls}$ (Section 3.1), and $\mathcal{L}_{dfl}$ — are combined either by fixed, manually-set gains, or by a learned alternative.

Fixed gains: $\mathcal{L}_{total} = g_{box}\mathcal{L}_{box} + g_{cls}\mathcal{L}_{cls} + g_{dfl}\mathcal{L}_{dfl}$, with g_{box}=7.5 and g_{dfl}=1.5 fixed throughout (YOLOv8's defaults, tuned for unmodulated BCE), and $g_{cls} \in \{0.5, 2.0\}$ depending on configuration (Section 5.4).

Adaptive balancing (Kendall et al., 2018): each gain is replaced by a learned scalar $s_i = \log \sigma_i^2$ for $i \in \{box, cls, dfl\}$,

$$\mathcal{L}_{total} = \sum_i e^{-s_i}\mathcal{L}_i + s_i \tag{4}$$

with s_i initialized so $e^{-s_i}|_{t=0}$ exactly reproduces the fixed gains above, then updated by gradient descent jointly with the network weights. e^{-s_i} acts as a learned precision for term i; the $+s_i$ summand penalizes trivially inflating s_i to suppress a loss term's contribution.

4 Experimental Setup

All experiments use YOLOv8s (Jocher et al., 2023) trained on the COCO2017 (Lin et al., 2014) train split (118,287 images) and evaluated on the COCO2017 val split (5,000 images, 36,781 instances), with a fixed 15-epoch budget, batch size 16, image size 640, and the AdamW optimizer (Loshchilov

Table 1: mAP50-95 on COCO val2017, mean ± std over 3 seeds.

Configuration	mAP50-95
Stock floor	**0.408 ± 0.001**
Sin-SMFL, cls = 2.0 boosted	0.402 ± 0.000
Sin-SMFL, adaptive balance	0.394 ± 0.001
Sine, no class balance	0.379 ± 0.000
Sin-SMFL (proposed, cls = 0.5)	0.376 ± 0.002
Focal baseline	0.358 ± 0.001

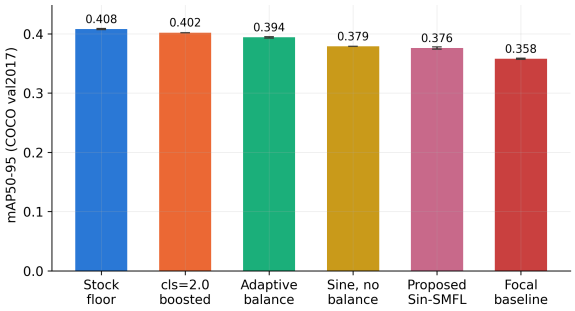

Figure 3: Overall ablation results (mean ± std, 3 seeds). Stock floor is the strongest configuration at this training budget.

& Hutter, 2019). Every configuration is run with three random seeds (0, 1, 2) for the aggregate results in Table 1; the per-class analyses (Sections 5.2, 5.6) use seed 1 consistently across all six configurations, so that per-class comparisons are not confounded by seed identity. The training-dynamics comparison (Section 5.7) uses one representative seed per configuration, which is not held constant across configurations there. We study six configurations spanning two axes of loss customization classification-loss modulation and per-class reweighting plus one learned alternative to manual gain tuning:

- **Stock floor**: no modulation, uniform class weights (the framework's unmodified BCE classification loss)
- **Focal baseline**: power-law modulation ($\gamma = 2$, i.e. classical Focal Loss), effective-number class weighting ($\beta = 0.999$)
- **Sine, no class balance**: sine modulation ($n = 2$), uniform class weights
- **Proposed (Sin-SMFL)**: sine modulation ($n = 2$), effective-number class weighting, default classification gain (cls $= 0.5$)
- **Sin-SMFL, cls $= 2.0$ boosted**: identical to Proposed, with the classification-loss gain manually rescaled to cls $= 2.0$
- **Sin-SMFL, adaptive balance**: identical modulation and class weighting to Proposed, with all three loss-term gains (box, cls, dfl) replaced by a learned per-term uncertainty weighting (Section 3.4), initialized to reproduce the default fixed gains exactly and adapted during training

5 Results

5.1 Overall ablation results

Figure 3 shows Table 1 as a bar chart.

The aggregate ranking alone does not show *why* each configuration performs as it does, because the configurations differ in multiple design choices. Section 5.2 therefore examines each design choice separately.

Table 2: Isolated paired comparisons, 80 classes each, all configurations at seed 1.

Comparison	Isolates	Mean Δ	Improved	Worsened	Tied	Sign-test p
(a) sine+uniform vs. none+uniform	sine modulation alone	-0.0287	3	77	0	< 0.0001
(b) sine+eff_num vs. sine+uniform (cls=0.5)	eff.-number balancing alone	-0.0011	26	51	3	0.0059
(c) cls=2.0 vs. cls=0.5 (sine+eff_num)	classification gain alone	$+0.0248$	79	1	0	< 0.0001
(d) sine+eff_num vs. power+eff_num (cls=0.5)	modulation type (sine vs. power)	$+0.0198$	77	2	1	< 0.0001
(e) adaptive balancer vs. manual cls=2.0	correction method	-0.0079	12	65	3	< 0.0001
(f) adaptive balancer vs. uncorrected default	adaptive vs. no correction	$+0.0170$	75	4	1	< 0.0001

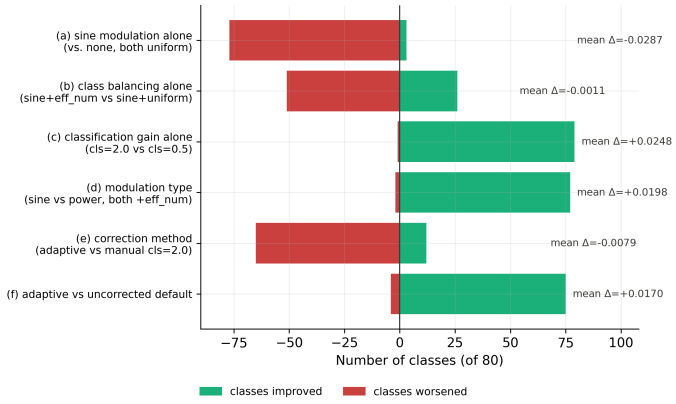

Figure 4: Isolated per-class effect of each design choice (seed 1, matched). Green bars: classes improved; red bars: classes worsened, out of 80.

5.2 Isolating individual design choices via paired per-class comparison

Because the six configurations differ along multiple axes simultaneously, we construct six paired comparisons, each holding every factor fixed except one, and compare mAP50-95 on all 80 COCO classes as matched pairs. All six configurations' per-class results are drawn from the same seed (seed 1), so that seed identity does not itself confound the isolation. For each pair, we report the mean per-class change, the number of classes that improve or worsen, and a two-tailed exact sign test. The null hypothesis is that a class is equally likely to improve or worsen. We treat the sign-test p-values as an informal consistency check rather than a formal significance test, since the 80 classes are evaluated from the same trained model and are not independent samples (see Section 6).

Figure 4 shows Table 2 as a diverging bar chart of improved/worsened class counts per comparison.

5.3 Sine modulation outperforms Focal Loss's power modulation, holding balancing and gain fixed

Comparison (d) in Table 2 examines the effect of the modulation function itself. Sine modulation improves mAP50-95 on 77 of 80 classes compared with classical power-law (Focal Loss) modulation. Both configurations use the same effective-number class weighting and classification gain. This is the largest and most consistent class-level effect in our study after the gain-correction result (c).It shows the contribution of the proposed modulation function while keeping all other loss components unchanged. It also directly tests the bounded-derivative motivation described in Section 3.2.

5.4 Sine modulation is broadly harmful when deployed with an uncorrected gain

Comparison (a) shows the opposite pattern when sine modulation is compared against no modulation at all (both with uniform class weights, both at the default cls $= 0.5$ gain): 77 of 80 classes get worse. Read together with Section 5.3 that the two results are not contradictory because they address different questions.Sine modulation is a better modulation function than the power-law alternative (Section 5.3). However, any modulation reduces the classification-loss magnitude compared with plain BCE. As a result, the framework's default gain, which is tuned for unmodulated BCE, under-

Table 3: Stock floor mAP50-95 under classification-gain sweep (1 seed).

cls gain	0.3	0.5 (default)	1.0
Stock floor	0.401	0.408 ± 0.001	0.411

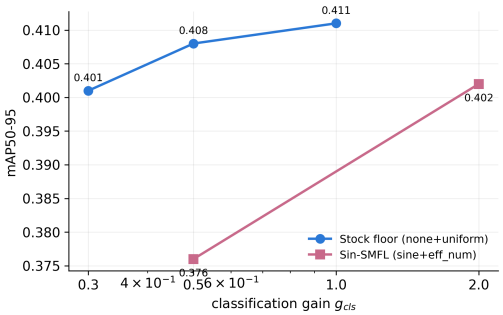

Figure 5: Classification-gain sensitivity: stock floor vs. Sin-SMFL. Sin-SMFL's slope is markedly steeper, and the two lines never cross within the tested range.

weights the classification loss when modulation is introduced. Deploying sine modulation without correcting for this issue harms performance across many classes.

Correcting the gain manually (comparison c) recovers this almost completely: 79 of 80 classes improve when cls is rescaled from 0.5 to 2.0, closing Sin-SMFL's aggregate gap to the stock floor from 0.032 to 0.006 (Table 1).

Controlling for asymmetric tuning. We tuned this gain correction specifically for Sin-SMFL to test whether the stock floor also has room for improvement through gain tuning. This is important because such an improvement could weaken the comparison above. We therefore also sweep the stock floor's classification gain, with cls $\in \{0.3, 1.0\}$, while keeping its modulation and class weighting fixed at the stock settings (none, uniform).

The stock floor improves with a higher gain, but only slightly: $+0.010$ across a $3.3\times$ gain range ($0.3 \rightarrow 1.0$). In comparison, Sin-SMFL improves by $+0.026$ across a comparable $4\times$ gain range ($0.5 \rightarrow 2.0$). The stock floor is roughly $2.5\times$ less sensitive to this hyperparameter. This is consistent with the explanation above: when modulation does not reduce the loss magnitude, adjusting its weight provides less improvement. Critically, controlling for this does not close the residual gap: at its best tested gain (cls$=1.0$, mAP50-95$=0.411$), the stock floor still exceeds Sin-SMFL at its best tested gain (cls$=2.0$, mAP50-95$=0.402$) by 0.009 if anything slightly larger than the originally reported asymmetric gap of 0.006. We consider the classification-gain mismatch to be confirmed by this symmetric check. The remaining gap to the stock floor appears to be real rather than a result of uneven tuning.

5.5 Adaptive balancing improves broadly over the default, but underperforms manual tuning at the class level

The learned uncertainty-weighted balancer (comparison f) improves 75 of 80 classes relative to the uncorrected default, confirming the aggregate finding in Table 1 (0.394 vs. 0.376) is a broad effect rather than a few classes carrying the mean. However, comparison (e) shows it improves fewer classes (12) than it worsens (65) relative to the manually tuned cls $= 2.0$ configuration. A more one-sided pattern than the aggregate gap (0.394 vs. 0.402) alone would suggest. The adaptive balancer is a consistent improvement over doing nothing, obtained without any manual hyperparameter search, but it is not yet a class-level match for a correctly tuned manual gain.

Table 4: Mean per-class Δ mAP50-95 (sine+eff_num − sine+uniform, cls=0.5) by rarity cutoff, seed 1.

Cutoff	Mean Δ	Improved	Worsened
Bottom-5	+0.0224	3	2
Bottom-10	+0.0139	7	3
Bottom-15	+0.0090	9	6
Bottom-20	+0.0059	10	10
Bottom-25	+0.0027	11	14
Bottom-30	+0.0017	12	18

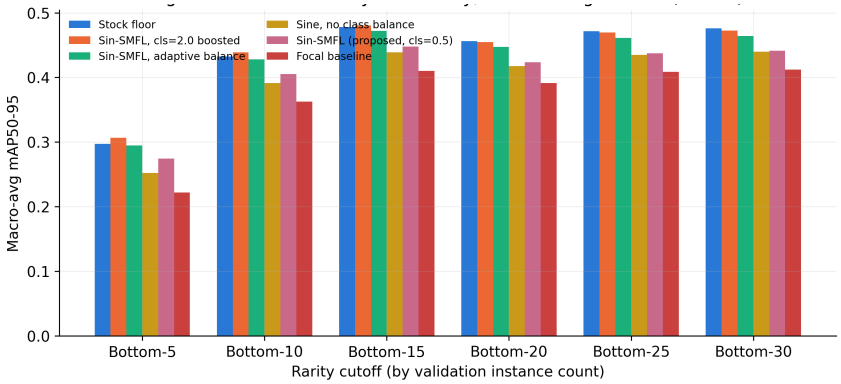

Figure 6: Performance by class rarity, all six configurations (seed 1). Ranking is preserved at every cutoff.

5.6 Class balancing: concentrated benefit for rare classes, no reduction in overall cross-class spread

We examine effective-number class balancing (comparison b) from two additional angles. The sign test itself is statistically significant ($p = 0.0059$) with more classes worsening than improving (51 vs. 26) but the mean effect is tiny (-0.0011) and, as the two analyses below show, its sign and magnitude depend heavily on which classes are examined.

Figure 6 shows Table 4 extended to all six configurations, not only the isolated pair.

Rarity-tiered effect. Restricting the same paired comparison to the N rarest classes, based on validation instance count, shows that the benefit is concentrated among the rarest classes and decreases as N increases:

This is consistent with the theoretical behavior of effective-number weighting, whose reweighting curve is steepest at the lowest sample counts. However, the crossover occurs earlier than initially expected. By the bottom-20 classes, the improved/worsened split is already even, and beyond this point, more classes worsen than improve, even though the mean delta remains positive. This positive mean is driven by a small number of larger positive outliers. We note that the `toaster` class (9 validation instances) has the largest positive delta (+0.130) and is the main driver of the bottom-5 result. With so few validation instances, a handful of detections can significantly change AP. We therefore treat this individual result as noise-sensitive, while the rarity-decay trend does not depend on it alone. We did not verify `toaster`'s delta against additional seeds; this remains an open check.

Cross-class spread. A separate question is whether class balancing reduces *inequality* of performance across classes, independent of the mean. We compute the standard deviation and coefficient of variation (CV) of per-class mAP50-95 across all 80 classes, seed 1:

Adding effective-number balancing (no_class_balance → proposed) does not reduce cross-class spread. The CV increases slightly, from 0.447 to 0.454, consistent with the earlier mixed-seed result (0.449 → 0.454). This result remains stable after applying the seed-matching correction described in Section 4. We expected class balancing to reduce cross-class inequality by giving more weight

Table 5: Cross-class spread of per-class mAP50-95, seed 1.

Configuration	Mean	Std	CV
Stock floor	0.408	0.167	0.410
cls = 2.0 boosted	0.403	0.169	0.421
Adaptive balance	0.395	0.169	0.427
Sine, no class balance	0.379	0.170	0.447
Proposed (sine+eff_num)	0.378	0.171	0.454
Focal baseline	0.358	0.168	0.470

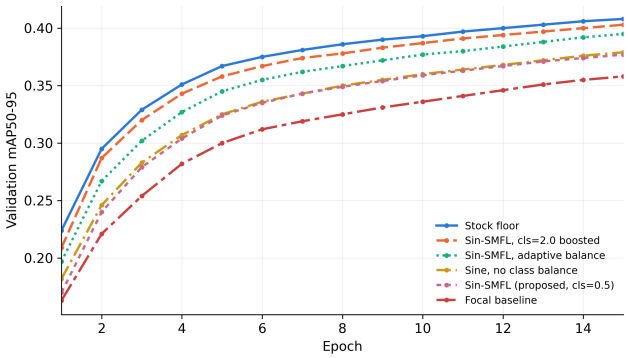

Figure 7: Training dynamics (1 representative seed per configuration, not held constant across configurations). Ranking is preserved from early training through epoch 15.

to rare classes. The data do not support this hypothesis. Table 4 and Table 5 describe different but non-contradictory effects of the same mechanism. Class balancing helps a small number of the rarest classes. At the same time, the significant but small sign-test result in Table 2 row b shows that slightly more classes worsen than improve overall. However, balancing does not change the overall shape of the per-class performance distribution.

5.7 Training dynamics

Figure 7 shows this comparison across all six configurations.

Comparing per-epoch validation mAP50-95 (one seed per configuration) shows all six configurations preserve the same relative ranking from early training through epoch 15, with no crossovers. The gap between the two closest configurations stock floor and cls $=$ 2.0 boosted narrows from 0.015 at epoch 1 to approximately 0.005–0.006 by epoch 10, and appears to plateau rather than continue closing at the same rate. We treat this as an open question rather than evidence of eventual convergence. Answering this question requires training beyond our fixed 15-epoch setting.

6 Limitations

All results use 15 epochs, not the usual 300. Modulated losses need more time to help hard and rare classes, so results may change with longer training. This is a short-budget study, not a claim about final performance.

The gain sweep for the stock model uses only two values, cls $=$ 0.3 and 1.0, from a single seed. It does not test cls $=$ 2.0, which is included for Sin-SMFL. The gains appear to slow as the classification gain increases, suggesting that the stock model is unlikely to surpass the tuned Sin-SMFL result, although this is not confirmed.

Author Contributions

Not disclosed for double-blind review.

ACKNOWLEDGMENTS

Not disclosed for double-blind review.

REFERENCES

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 794–803, 2018.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9268–9277, 2019.

Agrim Gupta, Piotr Dollár, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5356–5364, 2019.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLOv8. `https://github.com/ultralytics/ultralytics`, 2023.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11662–11671, 2020.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The iNaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8769–8778, 2018.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.