Accelerating Hyperparameter Tuning for YOLO-Based Endangered Sea Turtle Classification via an LLM-Agent Recommendation

Jong Won Baek [1] **Gwon Yul Jo** [1] **Tae Wan Goo** [1]

Abstract

Hyperparameter tuning for object detection is a time-consuming cognitive task that ML researchers face repeatedly: every candidate configuration demands a full training run, and conventional search algorithms routinely consume hundreds to thousands of hours per tuned model. We share a simple AI-assisted workflow that collapses this inner loop into a one-shot literature-driven recommendation from an LLM-based research agent (Nexus Science): given a task description, dataset properties, and a target model variant, the agent surveys $\geq$500 published papers and returns a single hyperparameter configuration in under one hour. Applied to six YOLO variants (v8x, v9e, v10x, 11x, 12x, and 26x) on a seven-species sea turtle classification task, the workflow yields an average mAP@0.5 improvement of $+0.089$ over Ultralytics defaults—larger than the gain from any architectural upgrade within the same family. Each per-model recommendation completes in 18–42 minutes, eliminating over 99% of the wall-clock cost reported for hyperparameter optimization (HPO) approaches studied in prior work on YOLO models. We release the full 13-step prompt, recommended configurations, and verification scripts so that other practitioners can reproduce or adapt the methodology with their preferred LLM-based research agent. Together, this work offers a concrete, reproducible case for accelerating hyperparameter tuning by replacing search-based HPO with a one-shot literature-driven LLM recommendation.

[1]BioNexus, 11F, Gwanggyo Business Center, 156 Gwanggyo-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Republic of Korea. Correspondence to: Jong Won Baek <jongwon.baek@bionexus.kr>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Sea turtles are among the most threatened marine taxa, with all seven extant species—*Caretta caretta*, *Chelonia mydas*, *Eretmochelys imbricata*, *Lepidochelys olivacea*, *Lepidochelys kempii*, *Natator depressus*, and *Dermochelys coriacea*—listed on the IUCN Red List (IUCN, 2024). Reliable, scalable identification from imagery is essential for population monitoring, by-catch reduction, and anti-poaching enforcement. Deep object detectors of the YOLO family have rapidly become the de facto standard for such tasks, owing to their favorable accuracy–latency trade-off and the steady release of improved variants from v8 through the recently announced v26.

Default hyperparameters for each new YOLO release are calibrated against general-purpose benchmarks such as COCO and are therefore frequently sub-optimal for fine-grained ecological datasets. Practitioners consequently rely on hyperparameter optimization (HPO) methods, which typically consume budgets one to two orders of magnitude larger than the cost of a single training run, regularly observed across YOLO-targeted HPO studies (Baek et al., 2026). Recent progress in large language model (LLM) agents suggests an alternative: rather than searching the hyperparameter space empirically, an agent can reason over the published literature to extract configurations that worked for similar models on analogous tasks, and propose a small set of high-quality candidates. Unlike fixed search algorithms, such a workflow also improves naturally over time at no additional algorithmic cost, as the literature grows richer and LLM-based research agents become more accurate and robust. In this paper, we use Nexus Science, an LLM-based research agent, to produce such recommendations, and we benchmark them across six YOLO variants on the seven-species sea turtle dataset.

Our contributions are as follows:

- **A step-by-step methodology applicable across LLM-based research agents.** We replace the conventional HPO inner loop—one full training run per candidate—with a single literature-driven recommendation, structured as a 13-step prompt that other practitioners can reproduce or adapt with their preferred agent (we use

Nexus Science) in a few hours.

- **Improved mAP performance accessible to non-AI domain experts.** The workflow lets practitioners unfamiliar with deep-learning engineering—such as ecologists or conservation researchers—obtain meaningful detection improvements over the official defaults without designing or running a hyperparameter search themselves.
- **Orders-of-magnitude reduction in tuning wall-clock time.** Every recommendation completes in under one hour per model, shortening wall-clock tuning time by several-fold up to several hundred-fold versus conventional search-based HPO of YOLO models.

2. Related Work

2.1. The YOLO Family

The YOLO family has continued to evolve since v8 (Jocher et al., 2023) through architectural refinements—anchor-free heads, end-to-end training without NMS (Wang et al., 2024a), programmable gradient information (Wang et al., 2024b), attention-centric backbones (Tian et al., 2025), and improved feature aggregation (Khanam & Hussain, 2024). We benchmark v8, v9, v10, 11, 12, and v26; their fine-tuning recipes remain broadly similar, motivating a unified treatment of hyperparameter recommendation.

2.2. Hyperparameter Optimization for YOLO

Hyperparameter tuning for YOLO has progressed from manual adjustment of the $\sim$30 settings exposed by Ultralytics (Jocher et al., 2024b) to algorithmic search (grid, random (Bergstra & Bengio, 2012), Bayesian (Snoek et al., 2012), or genetic), which replaces the human but not the cost: every candidate requires a full training run—"hundreds or thousands of GPU hours" for evolution (Jocher et al., 2024a) and "several hours to days" for the general tuner (Jocher et al., 2024b). YOLO-targeted HPO studies (Gill & Rajendran, 2026; Soltanikazemi et al., 2023; Sozol et al., 2025) confirm this scale but rarely report the HPO wall-clock itself; the closest analogue, Baek et al. (2026), reports a 300-iteration random search over 16 YOLO11s hyperparameters on a structurally similar invasive-turtle dataset and the same workstation (1,379 h), which we use as our concrete wall-clock reference.

We explore a third route: rather than sampling candidates on the target dataset, we use an LLM-based research agent (Nexus Science) to recover hyperparameter values that already worked in analogous published studies, replacing the search loop with a one-off automated literature survey.

2.3. LLM Agents for Scientific Workflows

LLM-based agents have been deployed across scientific workflows including literature review, hypothesis generation, code synthesis, and experimental design. We extend this paradigm to hyperparameter recommendation: given task and dataset descriptions, the agent surveys published configurations and synthesises a candidate setting via agentic reasoning.

3. Methodology

3.1. Problem Setting

Each YOLO variant is fine-tuned on the sea turtle dataset under a hyperparameter configuration that controls the optimizer, learning-rate schedule, augmentation magnitudes, loss weights, image size, batch size, and roughly twenty other settings. Every training run produces a model whose mAP@0.5 on the held-out test split serves as our quality metric. Conventional hyperparameter optimization searches the configuration space by repeatedly running this training pipeline, scoring each candidate, and selecting the highest-scoring one. We instead query an LLM agent that takes a task description (dataset statistics, hardware constraints, target architecture) and returns a single recommended configuration in minutes; we then train one model with that configuration and compare its mAP@0.5 against that of the official Ultralytics default.

3.2. Dataset and Augmentation

We use the seven-species sea turtle dataset curated by Baek et al. (2024), comprising 1,627 images (*Caretta caretta*, *Chelonia mydas*, *Eretmochelys imbricata*, *Lepidochelys olivacea*, *Lepidochelys kempii*, *Natator depressus*, and *Dermochelys coriacea*) collected from iNaturalist and Google Images and manually verified using species-specific morphological characteristics. The images vary substantially in pose, occlusion, illumination, and underwater visibility, reflecting realistic monitoring conditions, and exhibit a 12.4:1 inter-class imbalance between the most and least frequent species. The data is partitioned 64/16/20 into training (1,037), validation (258), and test (332) splits, following the original release.

To mitigate overfitting we apply two complementary augmentation pipelines during training: (i) Albumentations, which adjusts image hue, saturation, and brightness, and applies translation, scaling, and flipping; and (ii) Mosaic Augmentation, which composes four training images into a single composite to enrich the spatial distribution of bounding boxes.

3.3. Models and Training Setup

We evaluate the largest variant of each YOLO version: YOLOv8x, YOLOv9e, YOLOv10x, YOLO11x, YOLO12x, and YOLO26x, each initialized from the official pretrained COCO weights and fine-tuned on our dataset. All experiments are conducted on a single workstation running Rocky Linux 8 with two Intel Xeon Gold 6326 CPUs, 512 GB ECC DDR4 RAM (8×64 GB REG.ECC modules), and a single NVIDIA RTX A5000 GPU (24 GB). The training stack comprises Python 3.8.2, CUDA 12.2, cuDNN 8.9.3, and PyTorch 2.0.1—an environment identical to that of Baek et al. (2024; 2026), so that hardware and software are held constant when our wall-clock numbers are placed alongside their reported figures.

3.4. Nexus Science Agent

Nexus Science is an LLM-based research agent built on top of OpenClaude with a scientific harness engineering layer and GLM-5.1 as its reasoning backbone. We provide the agent with a structured prompt specifying the task (seven-class detection, fine-grained marine species, modest dataset size), the available hardware, and the target model variant. The agent then executes an automated literature survey across at least 500 papers, extracting hyperparameter configurations that yielded strong results on analogous tasks, and outputs a single recommended configuration per model along with a per-parameter justification and a predicted mAP improvement. We record the wall-clock time from prompt submission to recommendation availability separately from the subsequent training time. The harness engineering supporting the agent is described in Appendix A, and the full 13-step prompt structure is given in Appendix B.

3.5. Evaluation Protocol

We compare default and recommended configurations using mAP@0.5 on the test split, reporting the per-model improvement (ΔmAP) and the family-wise mean. We additionally report the wall-clock time of each recommendation and place it alongside the random-search timing of Baek et al. (2026) on the same workstation; because their dataset differs from ours, this is treated as an order-of-magnitude comparison rather than a strict head-to-head benchmark.

4. Results

We analyze the results along three axes: the magnitude of the mAP improvement (Table 1), its size relative to architectural upgrades (Figure 1), and the wall-clock cost of obtaining a recommendation relative to conventional HPO (Table 2).

Table 1. mAP@0.5 of each YOLO variant under default vs. Nexus-Science-recommended hyperparameters on the seven-species sea turtle dataset. Bold indicates the family-wise mean.

Model	Default	Tuned	ΔmAP
YOLOv8x	0.680	0.802	+0.122
YOLOv9e	0.665	0.820	+0.155
YOLOv10x	0.688	0.727	+0.039
YOLO11x	0.752	0.806	+0.054
YOLO12x	0.738	0.773	+0.035
YOLO26x	0.670	0.798	+0.128
Mean	**0.699**	**0.788**	**+0.089**

4.1. mAP Improvements

Every model exhibits a positive improvement, with gains from $+0.035$ (YOLO12x) to $+0.155$ (YOLOv9e). The mean of $+0.089$ mAP is a 12.7% relative gain over the average default (0.699) and is robust across the family—positive for both the earliest (v8) and most recent (v26) variants tested—suggesting the agent identifies configuration patterns that generalize across the YOLO line rather than overfitting to a particular architecture.

4.2. Tuning vs. Architectural Upgrades

Figure 1 contrasts architectural-upgrade gains (panel a, relative to YOLOv8x default) with tuning gains on a fixed architecture (panel b). The mean upgrade gain is $+0.023$ and the best single step (YOLO11x, $+0.072$) falls below the mean Nexus-Science gain of $+0.089$, whereas every recommendation improves over its own default. On this dataset, tuning a fixed architecture thus yields a larger improvement

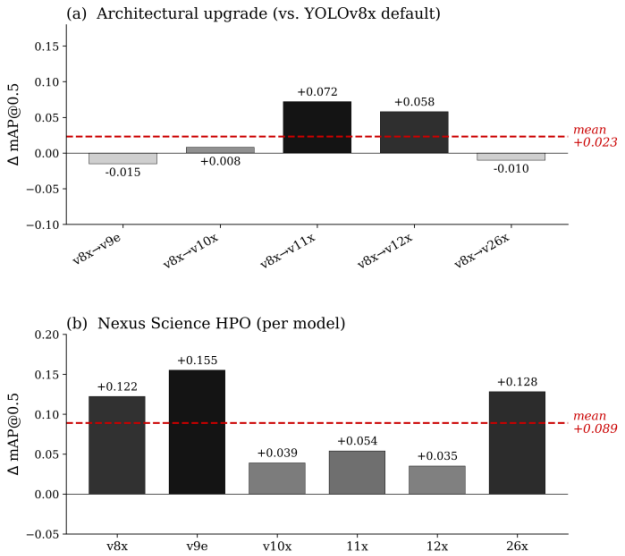

Figure 1. Tuning vs. architectural upgrades. (a) Default mAP@0.5 of newer variants relative to YOLOv8x (mean $+0.023$). (b) Per-model gain from recommended hyperparameters; all six positive, mean $+0.089$.

than upgrading to any newer YOLO version; under a fixed compute budget, hyperparameter recommendation is the more productive use of resources.

4.3. Wall-Clock Cost of Tuning

Wall-clock cost shows the most striking gap (Table 2): ≈30 minutes per recommendation against 1,379 hours for the only directly comparable prior HPO budget (Baek et al., 2026); other studies report only post-HPO training times and the Ultralytics guides only qualitative scales. The conclusion that literature-driven recommendation is cheaper than exhaustive search is therefore robust, even though a strict point-by-point comparison is not possible.

Table 2. Per-model wall-clock cost of obtaining a tuned configuration: our measured Nexus Science recommendation time per variant, against the ≈100 h lower bound from the Ultralytics guides ("hundreds of GPU hours", Jocher et al., 2024a) and the 1,379 h 300-iteration random search of Baek et al. (2026) on the same workstation. The last column lists the wall-clock reduction "vs Baek / vs Ultralytics".

Model	Nexus Science	Baek et al., 2026 / Ultralytics	Reduction (B/U)
YOLOv8x	18m 9s		99.98 / 99.70
YOLOv9e	35m 13s		99.96 / 99.41
YOLOv10x	42m 20s	1,379 h /	99.95 / 99.29
YOLO11x	25m 55s	≈100 h	99.97 / 99.57
YOLO12x	24m 17s		99.97 / 99.60
YOLO26x	37m 49s		99.95 / 99.37

5. Discussion

Across the six YOLO variants, the recommendations raise mAP@0.5 by $+0.035$ to $+0.155$ (family-wise mean $+0.089$, a 12.7% relative gain). Notably, the three weakest defaults (YOLOv9e, YOLO26x, YOLOv8x) receive the three largest gains, and the size of each gain tracks the headroom left by its default: the lower the starting point, the more the recommendation recovers, mainly by supplying the dataset-specific adjustments the generic defaults lacked. Set against the architectural axis (Figure 1), the mean upgrade gain relative to the YOLOv8x default is only $+0.023$, and even the strongest newer variant under defaults (YOLO11x, $+0.072$) sits below the mean recommendation gain of $+0.089$; on this dataset, the choice of hyperparameters shifts mAP more than the choice of YOLO generation.

A second contrast emerges from Table 2: each recommendation was completed within 18–42 minutes per model—approximately 3 hours across all six variants combined—whereas the 300-iteration random search reported by Baek et al. (2026) required 1,379 hours on the same workstation, indicating a substantially faster optimization process.

A further advantage of this approach is that it improves naturally over time. Unlike search-based HPO methods—which must re-incur compute for every new (dataset, architecture) pair—a literature-driven agent inherits the cumulative results of the field at no additional cost. As more empirical studies and HPO reports continue to accumulate for each YOLO variant, the agent's source material grows richer and its recommendations become more accurate and robust, making the workflow progressively more powerful as the community publishes—a compounding benefit unavailable to traditional search-based HPO. As a further synergy, editing only the task, dataset, and target model specified at the front of the prompt extends the same workflow to other models, datasets, and detection problems.

6. Limitations and Future Work

Our results are an encouraging first demonstration with several directions to strengthen the evidence: since prior YOLO-HPO studies used different datasets, our wall-clock comparison is an order-of-magnitude reference rather than a head-to-head benchmark; comparing conventional HPO algorithms on the same dataset and hardware with repeated seeded runs, and extending to more datasets and model families, are promising next steps.

The workflow also has room to grow. A recommendation may not always guarantee an improvement on the first attempt (Appendix E); in our runs, re-running recovered the configurations in Table 1, and studying this sensitivity and the non-improving cases is valuable. The survey may also inherit biases overrepresented in the YOLO literature, and we instantiate it only with Nexus Science; examining this bias and validating other agents are natural extensions.

7. Conclusion

This study replaced the conventional HPO search loop with a literature-driven LLM-agent recommendation, improving mAP@0.5 by $+0.089$ on average across six YOLO variants in only 18–42 minutes per model, tens to hundreds of times faster than search-based HPO. More broadly, literature-driven recommendation is a practical alternative to search-based HPO when compute or expertise is limited, and by lowering that barrier the workflow can support the research and conservation efforts of ecologists and other domain experts.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning through more efficient hyperparameter optimization. Our workflow lowers the compute and expertise bar for fine-grained ecological detection, which may broaden access to deep-learning-based wildlife monitoring for under-resourced research groups and conservation practitioners.

References

Baek, J.-W., Kim, J.-I., and Kim, C.-B. Deep learning-based image classification of sea turtles using object detection and instance segmentation models. *PLoS ONE*, 19(11): e0313323, 2024. doi: 10.1371/journal.pone.0313323.

Baek, J.-W., Kim, J.-I., Mun, M.-H., and Kim, C.-B. Hyperparameter optimization to enhance the performance of deep learning models for the early detection of invasive turtles in Korea. *Scientific Reports*, 16:7561, 2026. doi: 10.1038/s41598-026-37636-2.

Bergstra, J. and Bengio, Y. Random search for hyperparameter optimization. *Journal of Machine Learning Research*, 13(1):281–305, 2012.

Gill, M. U. and Rajendran, P. Hyperparameter optimization of YOLO using differential evolution, multi-fidelity optimization, and Bayesian optimization. *Scientific Reports*, 2026. doi: 10.1038/s41598-026-47827-6.

IUCN. The IUCN red list of threatened species. version 2024-2. https://www.iucnredlist.org, 2024.

Jocher, G., Chaurasia, A., and Qiu, J. Ultralytics YOLOv8. https://github.com/ultralytics/ultralytics, 2023.

Jocher, G., Munawar, M. R., and Mattioli, F. Hyperparameter evolution for YOLOv5. Ultralytics Documentation, https://docs.ultralytics.com/yolov5/tutorials/hyperparameter_evolution/, 2024a.

Jocher, G., Raimbekov, M., Yasin, M., Munawar, M. R., Qiu, J., and Derrenger, P. Ultralytics YOLO hyperparameter tuning guide. Ultralytics Documentation, https://docs.ultralytics.com/guides/hyperparameter-tuning/, 2024b.

Khanam, R. and Hussain, M. YOLOv11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Soltanikazemi, E., Dhakal, A., Diawara, B., Bhattarai, B., and Bunyak, F. Real-time helmet violation detection in AI city challenge 2023 with genetic algorithm-enhanced YOLOv5. *arXiv preprint arXiv:2304.09248*, 2023.

Sozol, M. S. S., Mondal, M. R. H., and Thamrin, A. H. Indoor fire and smoke detection based on optimized YOLOv5. *PLOS ONE*, 20(4):e0322052, 2025. doi: 10.1371/journal.pone.0322052.

Tian, Y., Ye, Q., and Doermann, D. YOLOv12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*, 2025.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., and Ding, G. YOLOv10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Wang, C.-Y., Yeh, I.-H., and Liao, H.-Y. M. YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*, 2024b.

A. Scientific Harness Engineering at v0.1.6

NexusScience v0.1.6 is a thin domain layer over the OpenClaude v0.1.6 coding-agent runtime. The science-domain work is organized into six engineering surfaces, summarized in Table 3.

Design constraint. The OpenClaude query loop, tool dispatcher, permission system, and hooks are not edited. All extensions register through hooks the runtime already exposes, the standard ToolDef interface, or middleware wrapping an existing dispatch boundary.

Table 3. The six engineering surfaces (S1–S6) that comprise the science-domain layer of Nexus Science v0.1.6. Each surface is implemented inside the OpenClaude runtime's standard extension points (hooks, ToolDef, or dispatch middleware) and ships with feature flags defaulting to off.

#	Surface	Code	Summary
S1	Gate chain	`src/query/gates/`	Eight gates compose on the assistant boundary via a shared BaseGate contract ({pass, nudge, escalate}). Read-only against state, skipped inside sub-agents, capped at 5 nudges per turn, escalates after 3 strikes. $<$50 ms p95.
S2	Science middleware	`src/nexus/middleware/`	injectSciencePrompt prepends the science-policy preamble at session start; scienceToolMiddleware canonicalizes paper identifiers (DOI/PMID/arXiv/URL) into a 12-hex key and tags fetched results for the citation allowlist.
S3	Tool registry	`src/nexus/tools/`	26 domain tools registered as ordinary ToolDefs, in five groups: paper pipeline (4), citation graph (3), persistent memory (8), experiment & data (8), reporting (3). Paper pipeline uses a three-tier extraction backend (pdftotext $\rightarrow$ pdf-parse $\rightarrow$ tesseract).
S4	Persistent memory	Brain* tools, NEXUS.md	Two surfaces: a content-addressed Brain archive that promotes session notes on graceful shutdown and reloads them on next start, plus NEXUS.md as primary project memory with CLAUDE.md fallback. Both survive auto-compaction.
S5	Bundled skills	`src/nexus/skills/bundled/`	Seven markdown skill files (research-init, literature-review, cross-paper-analysis, research-gap-finder, survival-analysis, ml-coding-checklist, generate-report) loaded by nexusSkillLoader and dispatched through the runtime's existing SkillTool.
S6	A/B eval harness	`eval/`	Content-addressed YAML fixtures (SHA-256 recorded per result), dispatch $\in$ {real, synthetic, echo} selected per-fixture, scoring declared as a metric or judge block, artifacts persisted at `eval/results/<track>-<v>/<model>/{a,b}/`. Every track flag ships default OFF.

Summary. S1–S6 account for the entirety of science-domain harness engineering at the v0.1.6 cut. The OpenClaude runtime below is consumed as a versioned dependency; no science-domain change at v0.1.6 requires a runtime patch.

B. Input Prompt for Hypothesis Generation

The first step of our pipeline elicits a hypothesis from Nexus Science: given the task, the dataset, and the target YOLO variant, it proposes a single hyperparameter configuration expected to improve mAP. The six prompts share the common framework in Appendix B.1 and differ only in the target model name and baseline-run directory.

B.1. Shared Prompt Framework

The original prompts are in Korean; the version below is translated and condensed. Each prompt specifies a role, a task, a text-only execution rule, the system environment, the input paths, the global constraints, the thirteen ordered STEPs, and the required OUTPUT FORMAT.

```
# ROLE
You are a Senior Computer Vision Researcher specializing in YOLO-based object
detection, underwater detection, literature-driven hyperparameter optimization,
mAP-based performance back-reasoning, and HW-aware training optimization.

# TASK
Analyse the trained <MODEL> result on the sea-turtle dataset and produce a
literature-grounded recommendation that improves mAP. Follow the 13 STEPs
below in strict order; do not skip stages or short-cut to a summary.

# TEXT-ONLY EXECUTION RULE
Never perform pixel-level vision analysis. Use only: annotations, bbox
statistics, image resolution metadata, dataset structure, training logs,
results.csv, confusion-matrix numbers, and paper metadata.

# SYSTEM ENVIRONMENT
OS: Rocky Linux 8 | CPU: Intel Xeon Gold 6326 x2 | RAM: 512GB ECC DDR4
GPU: NVIDIA RTX A5000 24GB
Stack: Python 3.8.2 | CUDA 12.2 | cuDNN 8.9.3 | PyTorch 2.0.1

# INPUT PATHS
Dataset: /Users/Desktop/<Dataset>
Baseline run: /Users/Desktop/<MODEL>/

# GLOBAL CONSTRAINTS
- At least 500 surveyed papers (single recommendation only)
- Performance-weighted analysis (higher-mAP papers carry higher weight)
- Reject paper-only or default-only justifications
- Recommend only hyperparameters that exist in <MODEL>
- All values must be executable under the listed HW

# STEPS (each STEP must be produced in order)
STEP 1.  Dataset state analysis
STEP 2.  Baseline <MODEL> result analysis (mAP, P, R, loss curves, bottleneck)
STEP 3.  <MODEL> hyperparameter inventory (only parameters that exist)
STEP 4.  Paper collection and metadata audit (>=500 papers)
STEP 5.  Per-parameter HP <-> mAP relationship (weighted, not frequency)
STEP 6.  Link dataset issues + baseline bottlenecks + paper patterns
STEP 7.  HW-aware feasibility (imgsz/batch ranges, workers, cache mode)
STEP 8.  Final recommended values (one value per HP)
STEP 9.  Per-parameter detailed justification
STEP 10. Final YAML
STEP 11. Predicted mAP improvement (single number)
STEP 12. Confidence assessment per HP
STEP 13. References (>=500 entries)
```

B.2. Notes on Prompt Design

The 13-step structure enforces an explicit dataset-state analysis before literature retrieval, performance-weighted aggregation of surveyed papers, and a hardware-feasibility pass under the 24 GB GPU memory budget. The minimum requirement of 500 surveyed papers per recommendation was chosen empirically.

C. Recommended Hyperparameters and Predicted Improvement

This appendix presents the default values alongside the recommended values for each YOLO variant. Experiments were conducted on the system described in Section 3 using the seven-species sea turtle dataset of Baek et al. (2024). Values that differ from the official Ultralytics defaults are shown in bold, and "auto" and "auto(SGD)" denote the same baseline.

Table 4. Complete hyperparameter recommendation for each YOLO variant. Default values are taken from the official Ultralytics configuration; bold entries indicate the agent's edits relative to that default.

Hyperparameter	Default	YOLOv8x	YOLOv9e	YOLOv10x	YOLO11x	YOLO12x	YOLO26x
Training settings							
imgsz	640	640	640	640	**1024**	**1024**	**1280**
batch	16	**8**	16	**8**	**8**	**4**	**8**
epochs	100	**150**	**150**	100	**200**	**200**	**150**
patience	100	**30**	**50**	**30**	100	**50**	**50**
cache	false	**true**	**ram**	**true**	**ram**	**true**	**true**
close mosaic	10	**20**	**15**	**20**	**30**	**30**	**15**
workers	8	**16**	8	8	8	8	**16**
dropout	0.0	**0.1**	0.0	0.0	0.0	0.0	0.0
freeze	null	null	null	null	null	null	null
Optimization							
optimizer	auto(SGD)	auto(SGD)	auto(SGD)	auto(SGD)	auto(SGD)	**AdamW**	auto(SGD)
lr0	0.01	**0.001**	0.01	**0.008**	**0.005**	**0.001**	**0.005**
lrf	0.01	**0.1**	0.01	0.01	0.01	0.01	0.01
momentum	0.937	0.937	0.937	0.937	0.937	0.937	0.937
weight decay	0.0005	0.0005	0.0005	**0.001**	0.0005	0.0005	0.0005
cos lr	false	**true**	**true**	**true**	**true**	**true**	**true**
warmup epochs	3.0	**5.0**	**5.0**	3.0	3.0	**5.0**	**5.0**
warmup bias lr	0.1	**0.05**	0.1	0.1	0.1	0.1	0.1
Loss							
box	7.5	**10.0**	7.5	7.5	7.5	7.5	7.5
cls	0.5	**2.0**	**0.8**	**0.8**	**2.0**	**1.5**	**1.0**
cls pw	0.0	0.0	**0.25**	0.0	0.0	**2.0**	**0.5**
dfl	1.5	**2.0**	1.5	1.5	1.5	1.5	1.5
Augmentation							
hsv h	0.015	0.015	0.015	0.015	0.015	0.015	**0.02**
hsv s	0.7	**0.5**	0.7	**0.5**	0.7	**0.5**	0.7
hsv v	0.4	0.4	0.4	**0.3**	0.4	0.4	0.4
degrees	0.0	**10.0**	0.0	0.0	**5.0**	0.0	**5.0**
translate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
scale	0.5	**0.6**	0.5	0.5	0.5	**0.3**	**0.6**
flipud	0.0	**0.2**	0.0	0.0	0.0	0.0	0.0
fliplr	0.5	0.5	0.5	0.5	0.5	0.5	0.5
mosaic	1.0	**0.8**	1.0	**0.8**	1.0	**0.5**	1.0
mixup	0.0	**0.15**	0.0	0.0	0.0	0.0	**0.05**
copy paste	0.0	**0.1**	0.0	0.0	**0.3**	**0.1**	**0.1**
copy paste mode	flip	flip	flip	flip	**mix**	flip	flip
auto augment	randaug	randaug	randaug	randaug	randaug	**auto**	randaug
erasing	0.4	**0.2**	0.4	**0.3**	**0.3**	**0.2**	0.4
multi scale	0.0	0.0	0.0	0.0	0.0	0.0	0.0

D. Training Convergence and Overfitting Check

To check for overfitting under the agent-recommended hyperparameters, Figure 2 plots per-epoch validation loss (box, cls, dfl components) for each of the six tuned models, with a red dotted line marking the kept checkpoint (best.pt, the epoch of maximum validation mAP@0.5). Across all six variants the validation loss declines smoothly with no late-epoch upturn, and every kept-checkpoint epoch sits on a non-rising segment. No overfitting is observed under the recommended configurations.

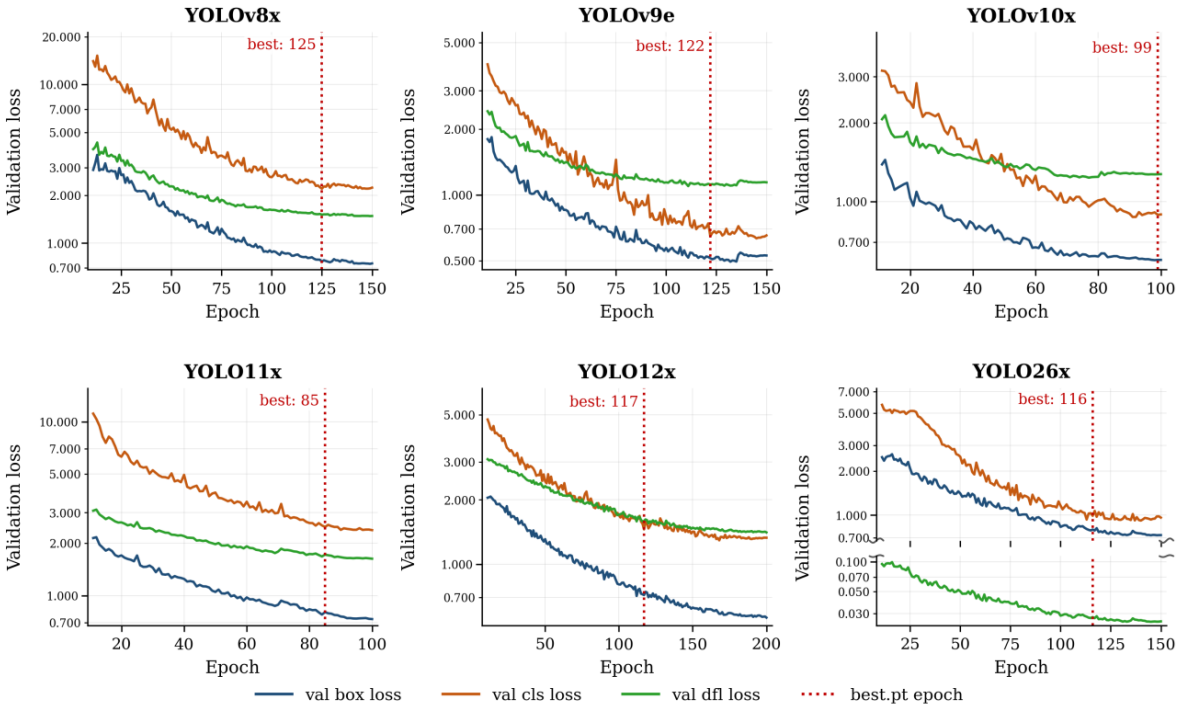

Figure 2. Per-epoch validation loss (box in blue, cls in orange, dfl in green) versus epoch for each of the six tuned YOLO variants. The red dotted vertical line in each panel marks the kept-checkpoint epoch (best.pt, the epoch of maximum validation mAP@0.5).

E. Cases Without Improvement: Training Instability and Sub-Default Convergence

We applied the tuning procedure to six models: four improved on the first pass, while the remaining two did not improve over the Ultralytics default, showing different kinds of problems—in one the training itself did not proceed properly, and in the other training proceeded well but the final performance still ended up below the default. We present them as Case 1 (YOLOv10x) and Case 2 (YOLO26x) respectively, and show the result under each table and figure. We note that these reflect first-attempt outcomes; the adopted configurations, obtained on re-running, are reported in Tables 1 and 4.

E.1. Case 1 — Training Did Not Proceed as Intended (YOLOv10x)

Case 1 (YOLOv10x) changed 17 settings from the Ultralytics default; Table 5 lists the default and tuned value of each.

Table 5. Case 1 (YOLOv10x): the 17 modified hyperparameters (default vs. tuned).

	imgsz	batch	epochs	patience	cache	close mosaic	freeze	cos lr	cls	dfl	hsv s	hsv v	degrees	scale	mosaic	mixup	erasing
Default	640	16	100	100	false	10	null	false	0.5	1.5	0.7	0.4	0.0	0.5	1.0	0.0	0.4
Tuned	736	10	200	30	true	20	0	true	0.8	2.0	0.6	0.35	5.0	0.45	0.7	0.05	0.3

Applying this configuration, the per-epoch validation loss in Figure 3 is not continuous: across the middle and later epochs several segments of each curve are broken, with stretches of missing logged loss values.

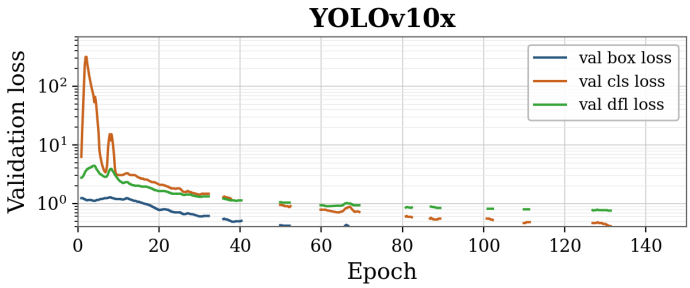

Figure 3. Case 1 (YOLOv10x): fused validation loss (log scale); the curves are discontinuous, with broken segments through the middle and later epochs.

These breaks in the recorded loss make the run unreliable, so this configuration was removed. Although it reached a best mAP@0.5 of 0.780 (above the Ultralytics default of 0.688), it was not adopted.

E.2. Case 2 — Stable Training but Below the Default (YOLO26x)

Case 2 (YOLO26x) changed 12 settings from the default; Table 6 lists the default and tuned value of each.

Table 6. Case 2 (YOLO26x): the 12 modified hyperparameters (default vs. tuned).

	imgsz	batch	epochs	optimizer	lr0	cos lr	cls	cls pw	degrees	mixup	copy paste	multi scale
Default	640	16	100	auto(SGD)	0.01	false	0.5	0.0	0.0	0.0	0.0	0.0
Tuned	1280	2	200	AdamW	0.002	true	2.0	0.8	10.0	0.1	0.1	0.5

Applying this configuration, all three loss components decreased smoothly and the model converged with no sign of overfitting. The tuned model's mAP@0.5 is compared against the default in Table 7.

Table 7. Case 2 (YOLO26x): detection performance (mAP@0.5) before vs. after tuning.

Model	Before (default)	After (tuned)	Δ
YOLO26x	0.670	**0.660**	-0.010 (below)

After tuning, the model's mAP@0.5 is 0.660—below the Ultralytics default of 0.670, a difference of -0.010.