Optimising the Measurable Layer: An Autonomous Agentic Workflow for BirdCLEF+ 2026 and Its Failure Modes

Venkata Pushpak Teja Menta[1]

[1]*Praxel Ventures Private Limited, India*

Abstract

We report the BirdCLEF+ 2026 entry of team Praxel: a code-competition system for 234-way acoustic species identification in Pantanal soundscapes. The official team result was public ROC–AUC 0.95087 and final private ROC–AUC 0.94216 (rank 397 of 4095 teams in the paged final leaderboard API); our post-reveal best single submission reached 0.94265 private but was not the official selected score. The contribution of this working note is therefore not a winning model recipe, but a reproducible account of *how* the campaign was run: 50 graded submissions over ten days, four trained model families, a multi-architecture ablation, and a hard monetary budget, all executed through an autonomous, self-pacing AI-assisted workflow of our own design. We set the scientific strategy, constraints, and experimental decisions, and directed an LLM agent loop (Claude Code) to carry out training launches, submission packaging, validation, budget accounting, and failure recovery.

We make four contributions. First, we document the agent architecture and a "commit-verify" gating discipline that took us from 14 silent no-score failures to *zero errored scoring kernels* across the 15-submission gated regime. Second, we give a controlled negative result: single CNNs, two-model ensembles, and confidence-gated "rescue" blends over a fixed public anchor span private 0.910–0.943, with no configuration reliably clearing the anchor. Third, we show a selection-time hazard of saturated plateaus: public and private ordering inverted inside our 0.95-public tier, so the best-private attempt could not be identified from public score alone. Fourth, we identify the workflow's central failure mode: a grade-seeking agent optimised the cheap-to-measure blend layer of a fixed public ensemble instead of the more valuable base-model layer. We end with a concrete checklist for future AI-assisted competition workflows.

Keywords

BirdCLEF, bioacoustics, agentic workflows, LLM agents, ensemble learning, model selection, leaderboard shake-up, negative results, reproducibility, domain shift

1. Introduction

BirdCLEF+ 2026, part of the LifeCLEF 2026 evaluation campaign, asks participants to identify 234 species (birds, insects, amphibians, and a few mammals/reptiles) from 5-second windows of soundscape audio recorded in Brazil's Pantanal, scored by macro ROC–AUC under a code-execution constraint (4 vCPU, no internet at inference) [1, 2]. The task is motivated by large-scale passive acoustic monitoring: a growing network of recorders can observe biodiversity continuously, but the resulting audio volume is too large for manual annotation. The public leaderboard rapidly saturated near 0.950: a large fraction of teams converged on a shared "public recipe"—Google Perch v2 embeddings, a Selective-State-Space temporal head (ProtoSSM), a distilled Sound-Event-Detection (SED) branch, and taxonomy-aware post-processing.

Our entry was distinctive in *how* it was run rather than *what* it produced. We set the strategy, targets, and hard constraints (e.g. "reach 0.955," "spend every daily slot," "stay under $27 of cloud GPU") and built an autonomous AI-assisted workflow to execute them: an LLM agent loop that, under our direction, ran training jobs, packaged kernels, submitted to the leaderboard, validated models, tracked budget, scheduled itself overnight, and recovered from failures. Over ten days we made 50 graded submissions, trained four model families, and conducted a multi-architecture ablation, with the loop waking on cron-like schedules and on job-completion events between our decision points.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany
pushpak@praxel.in (V. P. T. Menta)

Algorithm 1 Self-pacing agent loop (simplified). State (goals, prior results, hard constraints, budget) persists in a file-backed memory across context resets.

persist *state* ← {goals, constraints, budget, results}
loop ▷ woken by cron heartbeat or job-completion event
  $e \leftarrow$ Poll(monitors, leaderboard, training jobs)
  if $e =$ Idle **then**
    Sleep(t) ▷ t=1h idle; ~10 min near deadline
  else if $e =$ ModelReady **then**
    $m \leftarrow$ ExportONNX; ValidateHoldout(m) ▷ leak-free 15-file split
    $k \leftarrow$ BuildGuardedSidecar(anchor, m)
    CommitVerify(k) ▷ dry-run on visible test; inspect log
    if k clean $\wedge$ SlotsFree > 0 **then**
      Submit(k); Record(score)
  else if $e =$ BudgetSafe$\wedge$ new model warranted **then**
    LaunchTraining ▷ refuse if spend may breach cap
  ScheduleWakeup(t)

The campaign reached the public plateau (~0.950) but did not reliably clear it, and shook out near 0.942 on the private split. We believe documenting this workflow and its measured limits—the full blend-weight sweep, the validation-leakage analysis, the selection-time public/private inversion, and the reliability engineering—is more useful to the community than another model recipe, particularly given the LifeCLEF organisers' stated interest in "how participants used modern AI-assisted workflows … autonomous/agentic systems."

2. Related Work

BirdCLEF solutions across 2023–2025 converge on a stable recipe: large external pretraining (Xeno-Canto, prior-year competition data), iterative semi-supervised pseudo-labelling to bridge the focal-recording→soundscape domain shift, strong CNN/SED backbones, and large ensembles [10, 11]. The public anchor used in our campaign also depends on bioacoustic foundation embeddings from the Perch line of work [3, 4]. Our entry deliberately did not pursue this base-model direction deeply enough, and Section 4 quantifies the cost of that choice. Work on autonomous ML agents and LLM-driven experimentation is nascent; to our knowledge this is among the first end-to-end agentic accounts of a Kaggle-scale audio competition, including negative results and budget/reliability engineering.

3. The Agentic Workflow

We interacted with the workflow in natural language: we specified goals and constraints, and the agent loop we built translated them into shell actions, code generation, cloud-GPU jobs, and Kaggle API calls, persisting state in a file-backed memory so that our goals, prior results, and hard constraints survived context resets across the ten-day run. The design choices below—what to gate, what to guard, when to escalate compute—are ours; the loop is the instrument that carried them out.

Self-pacing loop. We had the loop schedule its own wake-ups (a cron-like mechanism with 1 h idle heartbeats, tightened to ~10 min near submission deadlines) and arm event monitors on long-running jobs (training completion, leaderboard scoring). To bound token cost, it distinguished cache-warm short waits (<5 min) from cache-cold long waits and chose intervals accordingly.

Commit-verify gating (the reliability core). The central discipline we imposed: every scoring kernel was first *committed* (a dry run on the visible test) and its log inspected *before* a leaderboard

slot was spent. This single discipline caught, pre-submission and at zero slot cost: (i) a dead wall-clock guard (`_remaining_notebook_time()` referenced but never defined); (ii) hidden-test-size timeouts (kernels sized against the tiny visible test that would exceed the limit on the larger rerun); (iii) a dataset mount-path bug (`/kaggle/input/<name>` vs `/kaggle/input/datasets/<owner>/<name>`) that silently disabled a sidecar; and (iv) a multi-label parsing error in our validation harness. Gating was adopted only after an early, un-gated exploratory phase had already lost submissions to silent rerun timeouts (e.g. five kernels on a single day that exceeded the hidden-test time limit and returned no leaderboard score). The effect is clean in the record: all 14 submissions that ever returned no leaderboard score predate the discipline (May 25–31), and *every one* of the 15 submissions after its adoption (June 1–3) scored—**zero errored scoring kernels in the gated regime**, a discipline we derived directly from our own earlier failures.

Guarded additive sidecars. New model contributions were appended to a known-good 0.950 anchor as a guarded cell: the anchor wrote its submission first, and the sidecar—wrapped in `try/except` plus an *absolute* kernel-time budget—could only improve the file or no-op. This made every probe incapable of erroring, timing out, or regressing below the banked floor.

Budget governance. We capped cloud-GPU spend at a $27 hard limit on Modal; the loop enforced this cap, declining launches that risked exceeding it, and recovered a detached run after an attached run was killed by a client-side connection timeout (`GRPCError: DEADLINE_EXCEEDED`)—switching to `modal run --detach` so the job survived disconnection. Total spend was ~$25 cloud GPU plus ~60 Kaggle GPU-hours.

4. Models and Experiments

Table 3 is the campaign's central artifact: a blend-weight sweep in which the *best* configuration we found (the final-hour "rescue" blend) exceeds the public anchor by only +0.0003 public / +0.0005 private—inside leaderboard noise—while most configurations merely *match* the anchor and weights above a calibrated threshold (e.g. w=0.65 for the strong ConvNeXt) regress catastrophically. No blend produced a reliable improvement. We report every graded probe in this June 1–3 anchor-blend sweep at full precision, including the regressions, rather than a best-per-model summary.

Table 1
Official competition context for the result reported in this note. The public leaderboard export and final leaderboard API were downloaded after leaderboard reveal on 24 June 2026.

Field	Value
Team and task	Team *Praxel*; BirdCLEF+ 2026 Kaggle code competition, macro ROC–AUC over 234 species.
Official selected score	Public ROC–AUC 0.95087; final private ROC–AUC 0.94216.
Official rank context	Public export: rank 549 of 4095 teams. Paged final leaderboard API: rank 397 in API order.
Best local submission after reveal	Submission timestamp 2026-06-03 00:03:47 UTC, "rescue t95 ConvNeXt-s1 confident non-Aves override", recorded in our export as 0.95063 public / 0.94265 private.
Selection status	The 0.94265-private submission was not the official selected team score. We therefore use 0.94216 for the official result and 0.94265 only as evidence for the public/private selection hazard.

Table 2
Reproducibility checklist for the anchor, sidecars, training data, and scoring notebooks. Exact public notebook and dataset slugs are included where they were available in the campaign logs; private Kaggle notebooks are named by their Kaggle slug.

Component	Specification
Fixed public anchor	"May27 Slot1 mirror-0950 EoS8+TAX_SMOOTHING ensemble", an unmodified mirror of the public EoS/0.946-family pipeline using Perch v2 embeddings [3, 4], ProtoSSM sequence modelling over twelve 5-second windows per 60-second soundscape, public `distilled-SED` scores, and genus/taxonomy smoothing. Earlier public ingredients used during reconstruction included `yashanathaniel/birdclef-2026-perch-v2-0-908`, `imaadmahmood/birdclef-2026-perch-v2-protossm-0-925`, `rishikeshjani/perch-onnx-for-birdclef-2026`, `perch-meta`, and `bc2026-distilled-sed-public`.
Submitted notebooks and packages	Private Kaggle scoring notebooks included `praxel/birdclef-2026-public946-robust`, `praxel/birdclef-2026-public946-headblend`, `praxel/birdclef-2026-advanced-v2`, and the final ConvNeXt rescue notebook family. All produced a Kaggle `submission.csv`; code-submission slots were spent only after visible-test dry-runs and log inspection.
Model checkpoints and exports	Perch union-head checkpoint `outputs/perch_head_union.pt` (Kaggle dataset `praxel/birdclef-2026-perch-head-union`); sidecar CNN checkpoints `best_fold{k}.pt` with ONNX exports `fold{k}.onnx`; B3 fold0, undertrained 20-epoch ConvNeXt-Tiny, and 24-epoch strong ConvNeXt/ConvNeXt-s1 rescue sidecars. Architectures used EfficientNet-style and ConvNeXt backbones [5, 6]; ONNX exports used ONNX Runtime [7].
Training data and labels	Official `train.csv`, `taxonomy.csv`, and `train_soundscapes_labels.csv`; 101k focal recordings; 1478 labelled soundscape windows expanded to 6650 five-second segments; 8122 high-confidence pseudo-labelled five-second windows. The class set was the official 234-species taxonomy; non-Aves blends targeted 72 classes while Aves columns retained the anchor.
Splits and leakage control	Multi-label stratified group folds by `audio_id`, 4 folds, seed 1086. A fixed 15-file soundscape holdout was stored in `outputs/holdout_split.json` and excluded via `HOLDOUT_FILES`; this corrected an earlier pseudo-label concatenation leakage risk.
Training configuration	Audio resampled to 32 kHz; five-second clips; 256-bin mel spectrograms resized to 256×256 for CNN sidecars. Default sidecar training used 20 epochs, batch size 64, Adam-style weight decay 10^{-4}, initial/final learning rates $2\times10^{-5}/10^{-4}$, maximum learning rate 5×10^{-4}, 5 warmup epochs, dropout 0.3, and best-fold export to PyTorch and ONNX. Modal [8] and Kaggle GPU jobs were used only for training; inference notebooks ran with internet disabled.
Pseudo-label procedure	Pseudo labels came from `praxel/bc2026-pseudo-labels/pseudo_labels_07.csv`. We retained rows with probability at least 0.7, mapped label indices through `taxonomy.csv`, capped each species at 100 windows, cut five-second segments from `train_soundscapes`, and excluded the holdout files before training.
Blend and post-processing	Ordinary sidecars rank-blended each non-Aves column into the anchor; Aves columns were unchanged. Rescue submissions only lifted non-Aves rows where sidecar confidence exceeded thresholds $t \in \{0.80, 0.85, 0.90, 0.92, 0.95\}$. All variants preserved the official 235-column file shape and used the same taxonomy/genus smoothing inherited from the anchor.

For a non-Aves class c and row r, ordinary sidecar submissions used

$$s_{r,c} = (1 - w)\ \mathrm{rank}(a_{\cdot,c})_r + w\ \mathrm{rank}(m_{\cdot,c})_r,$$

where a is the anchor, m is the sidecar output, and Aves classes retained the anchor values.

Table 3

Blend-weight sweep over the fixed public anchor (a mirror of the public "EoS"-family ensemble: Perch v2 embeddings + ProtoSSM Selective-State-Space head + `distilled-SED` + genus/taxonomy smoothing, the recipe shared by most teams at the ~0.950 public plateau, executed unmodified as our baseline). Each model family was rank-blended into the anchor over the 72 non-Aves classes at the listed weight w. Public/private are the actual macro ROC–AUC returned by the Kaggle leaderboard for each graded submission, at full precision (no post-hoc selection; regressions shown). The best blend (rescue t=0.95) exceeds the anchor by ~0.0005 private—within noise—most configurations match it, and high weights regress sharply, exposing the fragility of the blend layer.

Model family (sidecar)	w	Public	Private
Public anchor (mirror EoS8)	—	0.95035	0.94216
EfficientNet-B3 (focal CV 0.958)	0.15	0.94852	0.94201
" , class-gated	0.35	0.94606	0.94193
ConvNeXt-Tiny undertrained (val 0.73)	0.15	0.94716	0.93963
" (undertrained)	0.30	0.93698	0.93180
" (undertrained)	0.50	0.95035	0.94216
ConvNeXt-Tiny strong (val 0.92)	0.25	0.94533	0.93758
"	0.45	0.95035	0.94216
"	0.65	0.90959	0.90987
ConvNeXt + B3 two-model ensemble	0.10	0.95035	0.94216
Rescue non-Aves, t=0.92	—	0.94994	0.94250
Rescue non-Aves, t=0.95	—	**0.95063**	**0.94265**
V2-B0 sidecar (early probe)	0.20	0.92947	0.91499
Best of 50 graded submissions		0.95063	**0.94265**

Trained model families. On Kaggle P100 and Modal A100 we trained: EfficientNet-B3 and a B0 variant with soundscape pseudo-labels; SE-ResNeXt-26t; ECA-NFNet-L0; and ConvNeXt-Tiny (a 24-epoch "strong" run reaching held-out soundscape macro-AUC 0.82). Each was exported to ONNX and rank-blended into the anchor over non-Aves classes (the 72 of 234 classes where the Perch-based anchor is structurally weakest; Aves classes retain the anchor's values). The SE-ResNeXt-26t and ECA-NFNet-L0 sidecars were exercised in the earlier "jungchan-fork" multi-model probes (May 29–31), but those submissions returned no visible leaderboard score after competition close; we therefore exclude them from Table 3 and report only the verifiable June 1–3 single-model anchor-blend sweep.

The leakage wall (why local selection failed). The pseudo-label trainer concatenated the 1,478 *labelled* soundscape windows directly into its training set (`train = pd.concat([train, ss_df])`). Scoring any such model on those same windows is therefore pure memorisation: it reports inflated AUC (~0.92 "leaked" for B3) and would have *recommended submitting models that hurt the leaderboard.* We caught this by auditing the trainer source *before* trusting the validation gate—an instance where source-level scepticism, not metric-watching, prevented a wasted slot. We mitigated with a 15-file held-out soundscape split never seen by any model, on which the strong ConvNeXt scored a clean 0.82—correctly predicting that the blend would not clear the plateau.

The private shake-out. Because the entire pool is built from variants of one public recipe, it moved in lockstep on the private split: public ~0.950 → private ~0.942, a near-uniform −0.008 visible as the tight diagonal band in Figure 1. Because the pool contained no genuinely uncorrelated model, it had nothing that could resist this shift—a direct consequence of having optimised only the blend layer over a single public recipe.

Selection under a saturated plateau. The same root cause produced a second, subtler hazard at *final submission selection.* Within the 0.95-public tier, public and private ranking *inverted*: our two highest-*public* submissions (0.95087 and 0.95085) returned among the *lowest* private scores of the tier (both 0.94138), while our highest-*private* submission (the rescue blend, 0.94265) ranked only *third* on public (0.95063). The consequence is stark and counter-intuitive: the platform's default rule—absent a manual choice, score the two highest-*public* submissions—would have selected the 0.94138 pair, near the *bottom* of the private tier rather than the top. No signal available before the private reveal (best-public rank, or treating the tied submissions as interchangeable) identifies the best-private submission; the entire tier spans private 0.94138–0.94265, a ~0.0013 band indistinguishable from noise on the public

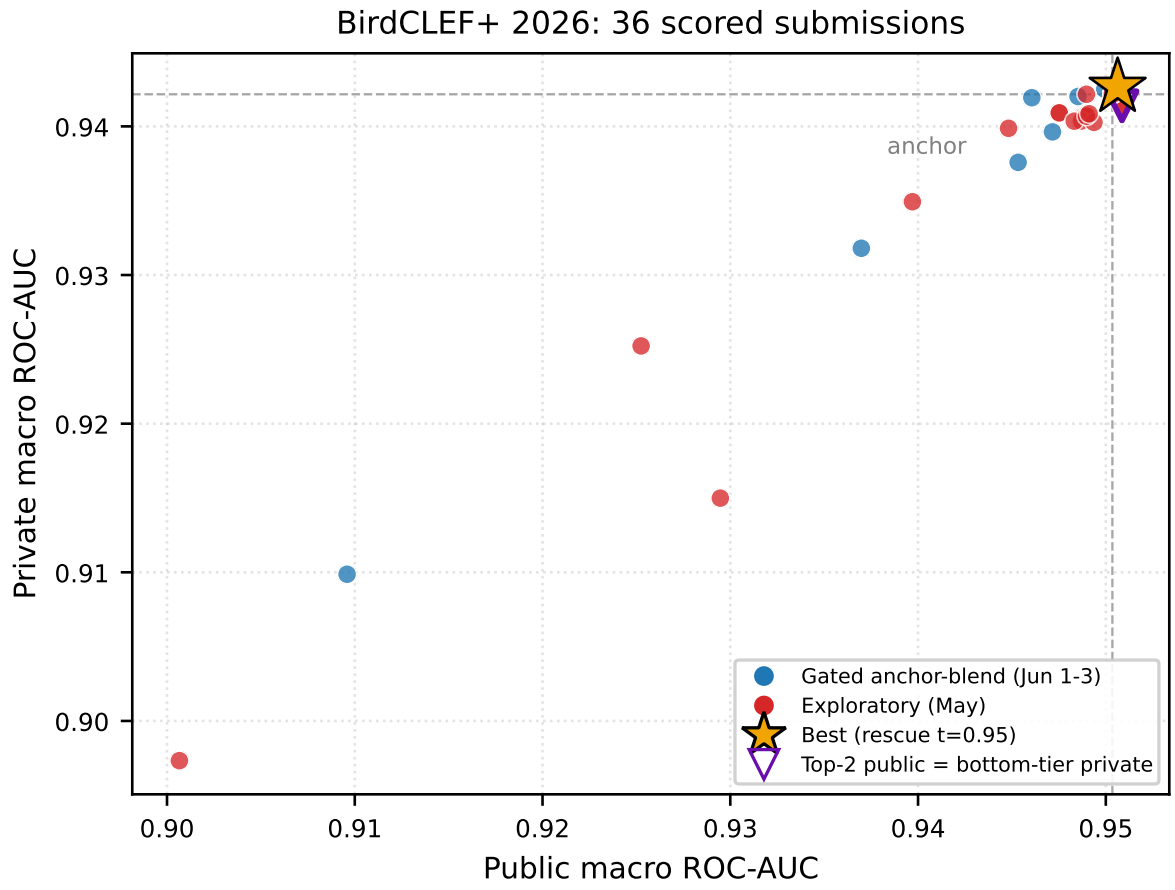

Figure 1: All 36 scored submissions across ten days (14 further kernels returned no leaderboard score). *No submission clears the public plateau by more than leaderboard noise*; the best (⋆, rescue t=0.95: 0.95063/0.94265) edges the anchor by ∼0.0005 private. The exploratory phase (red) and the commit-verify gated anchor-blend phase (blue) both saturate at the same public→private frontier. *Highlighted (open down-triangles, rightmost):* the two highest-*public* submissions carry among the lowest private scores, while the best-private submission (⋆) is only third on public—the selection-time inversion of Section 4. This visualises the central claim: a ten-day, breadth-first search of the *blend layer* never cleared the ceiling.

axis. This is the selection-time face of the lockstep shake-out: when a breadth-first search produces only recombinations of one public recipe, every submission collapses into a single public-tied, private-noise band in which the public leaderboard carries essentially *no* information about private rank, and even the platform default can land near the worst of the tier.

5. Lessons for Agentic ML Workflows

We state the central lesson as a principle and a corollary, then give the evidence and a checklist a practitioner can apply directly.

> ***The measurable-layer principle.*** A grade-seeking agentic workflow spends its effort on the layer of the solution it can *measure* most cheaply, not the layer that holds the most *value*; once that layer saturates it keeps polishing, because each probe still returns a legible number.
>
> ***Selection corollary.*** If the searched layer only recombines one public artifact, the candidate pool collapses into a single public-tied band whose *private* order the public score cannot resolve—so final selection has no signal, and the platform default can choose near the band's worst.

What the workflow did well. Reliability and discipline. Zero errored scoring kernels in the gated regime; a hard budget honoured; tireless overnight iteration; and—crucially—the *source-level scepticism* we built into the validation step, which caught a leakage trap (Section 4) that pure metric-watching would have walked into. These are the parts of competition engineering an AI-assisted loop automates almost perfectly: the mechanical, checkable, reliability layer.

The core failure: optimising the measurable layer. Our own ablation (Table 3, Figure 1) shows the blend layer was saturated: no weight, gate, or rescue threshold over a fixed public anchor cleared it by more than noise. Yet our search kept returning to that layer, because each probe is legible and immediately gradable, while we under-invested in the higher-variance, higher-payoff act of building stronger, *uncorrelated* base models—dedicated external-data pretraining and knowledge distillation, the directions established by published prior-year BirdCLEF work [10, 11]—work that is harder to "check" incrementally, which we repeatedly deferred. We argue this is a general bias of grade-seeking, AI-assisted workflows: effort flows to the most *measurable* surface, not the most *valuable* one. Our own measurements told us the surface was exhausted, and the loop kept polishing it regardless until we redirected it—and by then little time remained.

The selection corollary, measured. We did not anticipate the downstream cost. At the top of our pool public rank *anti-correlated* with private rank (Section 4): the best-private submission was only third on public, so the platform's best-public default would have selected near the *bottom* of the tier. The defence is upstream—hold at least one genuinely de-correlated submission—because no post-hoc selection recovers signal the public score never contained.

A checklist for agentic competition workflows. The principle and corollary reduce to five concrete rules, each of which we can trace to a specific episode in this campaign:

1. **Harvest prior art first.** Survey published prior-year solutions and their released artifacts *before* modelling, so effort starts at the valuable layer (base models) rather than the measurable one (blends).
2. **Commit-verify every scoring submission.** Dry-run on the visible test and inspect the log before spending a slot; this alone took us from 14 silent no-score failures to zero.
3. **Guard every probe.** Write the known-good anchor first, then wrap each contribution in `try/except` and an absolute time budget, so a probe can only improve or no-op—never error, time out, or regress below the floor.
4. **Impose a layer-budget.** Once a search layer returns no gain over a few probes, force effort to the next layer down instead of continuing to polish the saturated one.
5. **Diversify before you select.** Hold at least one genuinely de-correlated submission; on a saturated plateau public rank is not a selection signal, and no post-hoc cleverness recovers signal the public score lacks.

6. Conclusion

Using an autonomous, AI-assisted workflow of our own design, we ran a ten-day, 50-submission BirdCLEF+ 2026 campaign with strong reliability engineering and honest validation, and mapped the measured ceiling of public-ensemble recombination: a pool clustered within noise around 0.950 public / 0.942 private, with no blend configuration clearing the public anchor by more than noise. We offer the full blend-weight sweep, the validation-leakage analysis, the selection-time public/private inversion, and the workflow architecture as a reproducible case study. From the workflow's own trace we identify "optimise the measurable layer, not the valuable one"—and its selection corollary, that a saturated plateau hides its own best answer—together with a layer-budget mitigation, as a concrete, addressable failure mode of grade-seeking AI-assisted ML workflows.

Acknowledgments

We thank the LifeCLEF and Cornell Lab of Ornithology organisers, and the BirdCLEF community whose public kernels formed our anchor.

Declaration on Generative AI

During the preparation of this work, the author used Claude and Claude Code [9] for programming assistance, debugging, cloud-job orchestration, submission packaging, literature and forum review, experiment-log summarisation, and language polishing. The autonomous agent loop described in the paper is itself the object of study, and its use is therefore disclosed as part of the experimental method. After using these tools, the author reviewed and edited the content, verified the leaderboard results against Kaggle exports/API output, set the strategy and experimental constraints, drew the scientific conclusions, and takes full responsibility for the publication's content.

References

[1] L. Picek et al., *Overview of LifeCLEF 2026: AI Challenges for Biodiversity Understanding and Ecosystem Management*, International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF), Springer, 2026.

[2] S. Kahl et al., *Overview of BirdCLEF+ 2026: Acoustic Species Identification in the Pantanal, South America*, Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum, 2026.

[3] B. Ghani, T. Denton, S. Kahl, and H. Klinck, *Global birdsong embeddings enable superior transfer learning for bioacoustic classification*, Scientific Reports, 13, 22876, 2023.

[4] B. van Merriënboer, V. Dumoulin, J. Hamer, L. Harrell, A. Burns, and T. Denton, *Perch 2.0: The Bittern Lesson for Bioacoustics*, arXiv:2508.04665, 2025.

[5] M. Tan and Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, International Conference on Machine Learning, 2019.

[6] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A ConvNet for the 2020s*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[7] ONNX Runtime developers, *ONNX Runtime*, software, 2021–2026. Available: `https://onnxruntime.ai/`.

[8] Modal Labs, *Modal*, cloud execution platform, 2026. Available: `https://modal.com/`.

[9] Anthropic, *Claude Code*, agentic coding tool, 2026. Available: `https://www.anthropic.com/claude-code`.

[10] V. Sydorskyi, F. Gonçalves, *Tackling Domain Shift in Bird Audio Classification via Transfer Learning and Semi-Supervised Distillation: A Case Study on BirdCLEF+ 2025*, CLEF 2025 Working Notes, CEUR-WS, 2025.

[11] A. Miyaguchi, M. Gustineli, A. Cheung, *Distilling Spectrograms into Tokens: Fast and Lightweight Bioacoustic Classification for BirdCLEF+ 2025*, CLEF 2025 Working Notes, CEUR-WS, 2025.