Per-Block kNN Probing as a Layer Selector for Frozen Perch v2 Features: A Lightweight Sequence-Head Approach to BirdCLEF+ 2026

Sumit Yadav
Independent
sumit@astha.ai
https://github.com/rockerritesh

Abstract

We describe a BirdCLEF+ 2026 solution that scored Public LB 0.950 / Private LB 0.942 (rank 354/4084, top 8.7%, bronze zone) using a lightweight sequence head on top of frozen Google Perch v2 features blended with the publicly available Karnakbayev Power-Optimization ProtoSSM/SED chain. The methodological contribution is a cheap *per-block kNN probe* run once on 600 audio clips that ranks the 26 MBConv blocks of Perch v2 by their species-discriminative power. The probe correctly predicted that the penultimate block (MBConv_24) carries more transferable signal than the final block (MBConv_25), and that the trained attention pool's learned softmax weights would put block 24 first — which was independently confirmed after training (kNN ranking 0.648/0.595/0.555 vs learned weights 0.473/0.399/0.128 for blocks 24/25/21). We document a postmortem of nine final-week attempts to clear the public 0.950 plateau, including a useful negative result on single-pass pseudo-labelling that inflated out-of-fold AUC by +0.13 but regressed standalone LB by -0.009, and a checkpoint-version-overwrite gotcha discovered while integrating a DINOv2 ViT branch. The full session, including the AI-assisted engineering workflow, is documented in an open repository.

Keywords: bioacoustics, BirdCLEF, transfer learning, frozen features, attention pooling, ensemble methods, AI-assisted research

1 Introduction

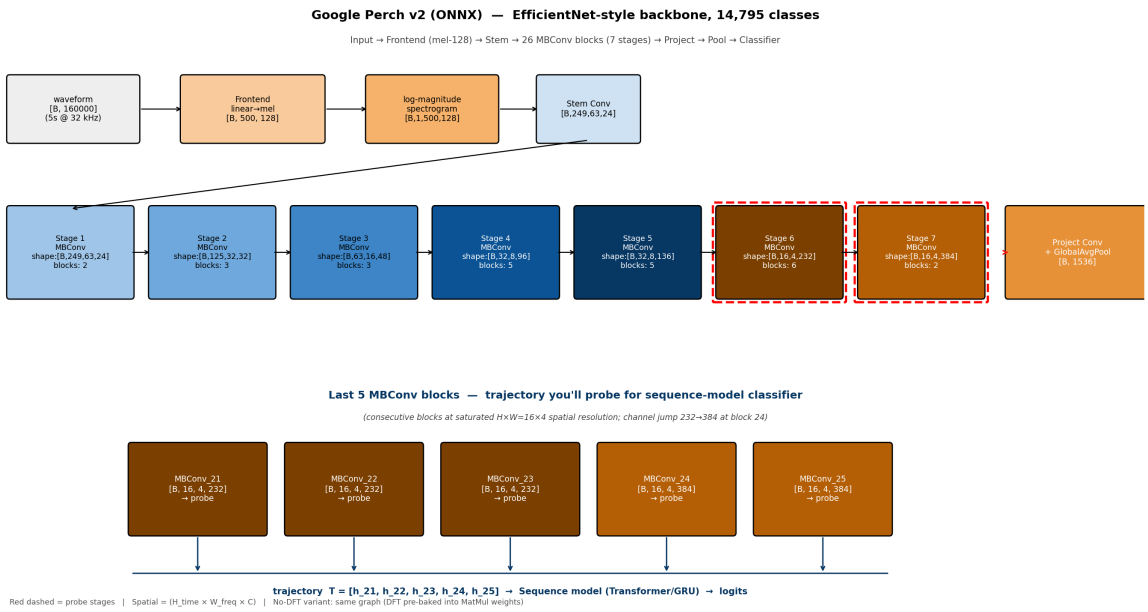
BirdCLEF+ 2026 Cornell Lab of Ornithology [2026] requires multi-label classification of 234 South-American Pantanal species across 5-second chunks of soundscape audio. The evaluation metric is macro-averaged AUC under a code-competition format (CPU-only inference, 90-minute runtime cap, internet disabled). Public and private leaderboards are split roughly 30/70 across the hidden test set.

The published baseline at LB 0.949 is the Karnakbayev Power-Optimization chain Karnakbayev et al. [2026], hideyukizushi [2026], which blends a ProtoSSM head on top of Google's Perch v2 Google [2026], Hamer et al. [2023] bird-vocalization classifier with Tucker Arrants' distilled SED Arrants [2026b]. Our contribution sits on the diversity side of that blend, with a lightweight (~430k-parameter) attention pool head reading frozen Perch v2 intermediate activations.

Two methodological choices distinguish this work:

- **A per-block kNN probe** run as a single Kaggle kernel on a 600-clip toy subset, ranking each candidate probe layer by 5-fold-CV 5-NN top-1 accuracy *before* any training. The probe predicted that the second-to-last MBConv block (block 24) is more useful for the BirdCLEF 234-class subset than the last one (block 25), because the final layer over-specializes to Perch's native 14,795-class softmax. The trained model's learned softmax weights independently confirmed the same ordering.
- **Honest documentation of the final-week ceiling**: nine attempts to clear the public 0.950 plateau, all landing at 0.950 or worse, including two useful negative results — pseudo-labelling and a ViT-backbone branch.

The full session log, every Kaggle kernel push (build script + generated notebook + metadata), and all rendered figures are available at `https://github.com/rockerritesh/BirdCLEF2026`.



80pt

Figure 1: Perch v2 architecture, recovered from the public no-DFT ONNX checkpoint. Stage 6 contains MBConv blocks 18–23 at $16 \times 4 \times 232$; Stage 7 contains MBConv blocks 24–25 at $16 \times 4 \times 384$. Our sequence head reads blocks 21, 24, 25 (red dashed outline).

2 Background: Perch v2 and the BirdCLEF Stack

Perch v2 is a Google-released audio classifier trained on 14,795 species (birds + amphibians + insects) via supervised contrastive learning followed by softmax fine-tuning Hamer et al. [2023]. The architecture (recovered by inspecting the public ONNX checkpoint) is an EfficientNetV2-style stack of 26 MBConv blocks across 7 resolution stages, plus a learned mel-spectrogram frontend and a final 1536-dimensional embedding projection (Figure 1).

The de facto strong baseline in BirdCLEF+ 2026 is a fusion of (a) the final 1536-d Perch embedding fed to a ProtoSSM head with rank-aware prior scaling, and (b) Tucker Arrants' distilled SED model (5-fold ensemble of EfficientNet-B0 trained via KLD against Perch outputs). The combined "Karnakbayev Power-Optimization" kernel, with TAX_SMOOTHING postprocessing, scores Public LB 0.949 Karnakbayev et al. [2026].

3 Per-Block kNN Probe

3.1 Method

The standard transfer-learning recipe in BirdCLEF reads Perch's final embedding (the 1536-d post-pool tensor). We hypothesized that the *last* MBConv block (the one just before final projection) might be over-specialized to the 14,795-class softmax, making it a suboptimal feature extractor for the 234-class BirdCLEF subset.

To test this cheaply we (i) augmented the Perch v2 ONNX graph to expose the outputs of 10 candidate MBConv blocks ($\{1, 4, 7, 12, 17, 21, 22, 23, 24, 25\}$) as additional graph outputs, then (ii) ran one Kaggle kernel that:

1. samples 30 species with 20 clips each (600 total) from `train_audio`;
2. forward-passes them through the augmented Perch v2 ONNX (one pass per clip);
3. spatially averages each probe output to a single channel vector;
4. per probe block, runs 5-fold-CV 5-nearest-neighbour top-1 accuracy on the 600-point feature cloud (chance $\approx 1/30 \approx 0.033$).

The entire probe runs in ≈ 15 min on Kaggle CPU and produces a ranking of probe layers *before* any model is trained.

Table 1: Per-block kNN-5 accuracy on 30 species × 20 clips. Block 24 outperforms block 25, contrary to the standard "use the final layer" convention.

Block	Stage	Dim	kNN-5
1	1	24	0.072
4	2	32	0.122
7	3	48	0.138
12	4	96	0.138
17	5	136	0.248
21	6	232	0.555
22	6	232	0.478
23	6	232	0.490
24	**7**	**384**	**0.648**
25	7	384	0.595

3.2 Results

Table 1 shows the per-block results. Two findings stand out: the biggest jump in discriminative power is the stage 5 → stage 6 transition (block 17 → 21, +0.30 kNN), and the penultimate block 24 beats the final block 25 (0.648 vs 0.595). The latter contradicts the default "use the last layer for transfer" assumption. Figure 2 shows per-species centroid trajectories through the last five blocks in joint UMAP space, illustrating the dispersion gained at block 24.

3.3 The probe predicts learned attention weights

We selected blocks $\{21, 24, 25\}$ as inputs to the sequence head (the top three individually informative blocks). A scalar-mix variant of the head *learns* a softmax over these three token projections during training. After training on focal recordings only, the 5-fold mean learned weights are $\alpha_{21} = 0.128$, $\alpha_{24} = 0.473$, $\alpha_{25} = 0.399$ — matching the kNN probe ordering both in rank and in approximate ratios. The 15-minute pre-training probe correctly predicted which layer the model would put the most weight on, before any gradient descent was performed.

We treat this as a useful, generalizable methodology: when working with a frozen backbone, a cheap kNN-probe sweep over candidate intermediate layers gives an inexpensive first-pass layer selector that the downstream head's weights will (probably) confirm.

4 Sequence Head

4.1 Architecture

At an input sequence length of $T = 3$ tokens (one spatially-pooled vector per probed block), most "sequence model" inductive biases pay for length they do not have — RNNs/LSTMs/Mamba get no benefit from recurrence at $T = 3$. We chose *attention pooling with a learnable [CLS] token* as the right inductive bias: each token is interpretable as one layer, attention weights are inspectable, and the parameter count is small (~430k).

The architecture is:

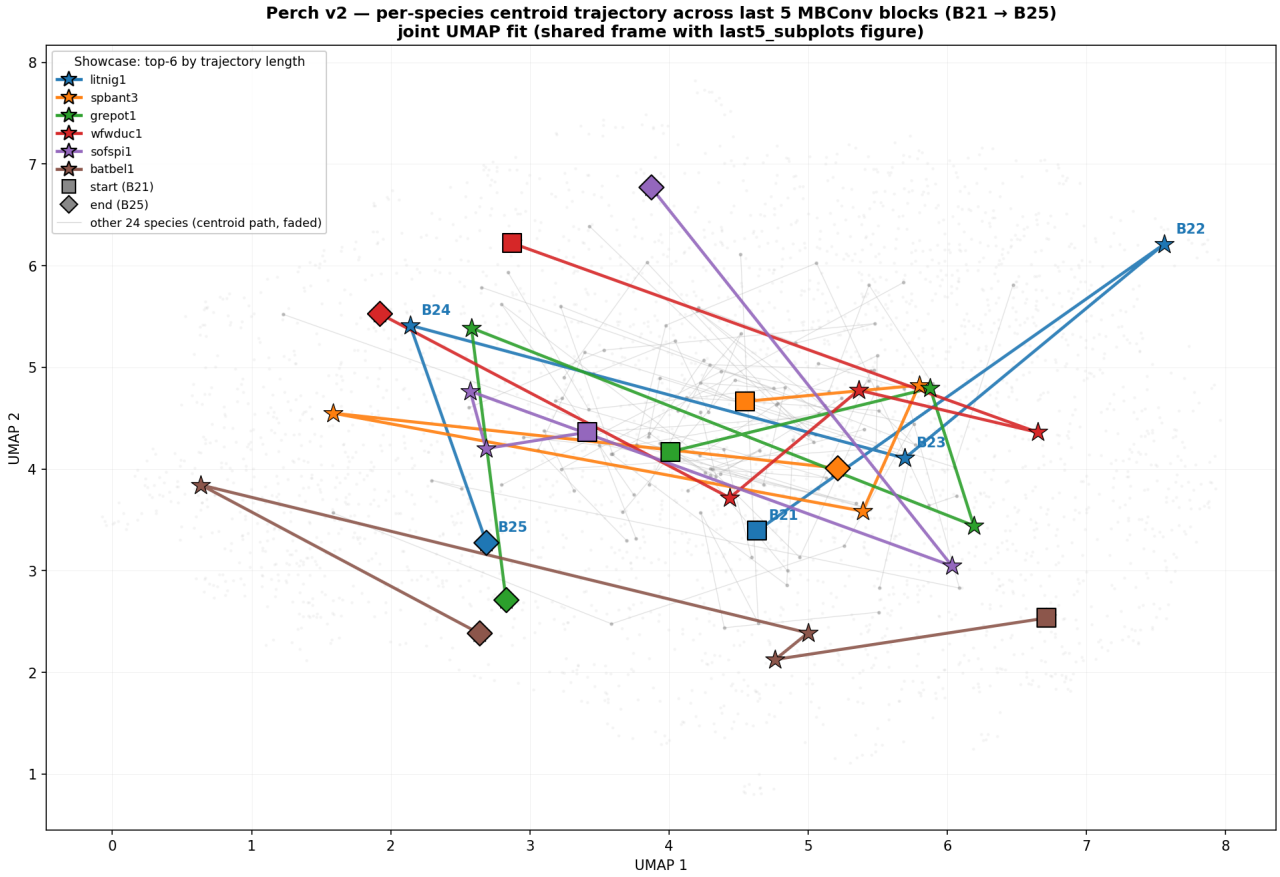
- Per-token projection $\text{Linear}(C \to 256) + \text{LayerNorm} + \text{GELU}$ to a common 256-dim space (three separate projections for $C \in \{232, 384, 384\}$).
- Learnable per-block embedding $E_b \in \mathbb{R}^{3\times 256}$ added to the projected tokens.
- Learnable [CLS] token prepended.
- One Pre-LN MHSA + FFN block (4 heads, $d = 256$, FFN= 512, dropout= 0.1).
- Linear $256 \to 234$ classifier reading the [CLS] output.

Loss is BCEWithLogits over multi-hot targets (primary + secondary labels).

4.2 Training

Features for the entire focal training set (~50k recordings, 3 random 5-s windows each = ~150k feature triples) are extracted in a one-shot GPU kernel, then cached as a 600 MB `.npz`. Subsequent training runs (4 variants × 5 folds) read the cache, training each fold in ≈ 5 min on Kaggle CPU.

-80pt-



80pt

Figure 2: Per-species centroid trajectory through MBConv blocks 21→25 in joint UMAP space. Six showcase species selected by total trajectory length; remaining 24 species shown as faded paths. Square marker = block 21 start; diamond marker = block 25 end; layer labels along the source species in red.

Five folds re-use a hash-by-filename split for the labeled soundscape data so each chunk receives exactly one out-of-fold prediction. The training loader is a WeightedRandomSampler with per-source batch fractions 75% focal / 15% labeled-soundscape / 10% pseudo-soundscape (the pseudo arm is disabled in the production v6 run; see Section 7.1). We use AdamW with cosine annealing, 50 epochs maximum with early stopping on fold soundscape macro-AUC, and feature-level Mixup ($\alpha = 0.2$) at batch construction time.

4.3 Standalone progression

Table 2 reports the standalone Public LB at each postprocessing step. The site/hour priors and Gaussian time smoothing follow the recipe published by Tucker Arrants in discussion 683791 Arrants [2026a] and applied to our backbone via a Bayesian log-odds boost $\text{logit} \leftarrow \text{logit} + \alpha \cdot \log(P(s \mid \text{site}, \text{hr})/P(s))$ with Laplace smoothing.

5 Meta-Ensemble Blend (eos7)

To submit competitively we forked the Karnakbayev power-opt kernel Karnakbayev et al. [2026] and integrated our model at a small weight:

- Model_73: Karnakbayev power-opt chain (ProtoSSM + SED, internal 0.60/0.40 blend), weight 0.93.
- Model_99: our v6 sequence head, weight 0.07.
- Output: weighted-add then TAX_SMOOTHING postprocessing (taxonomy-genus pull $\alpha = 0.15$, class pull $\alpha = 0.05$).

To fit the 90-minute CPU budget we dropped a third Karnakbayev sub-model (LB 0.928, weight 0.0327) that contributed only $\approx 3\%$ to the blend but consumed ≈ 30 min of CPU time. Figure 3 contrasts the standalone v6 pipeline with the eos7 meta-ensemble.

The eos7 kernel scored Public LB 0.950 (+0.001 over the Karnakbayev baseline) in ~65 min CPU.

Table 2: Standalone v6 sequence-head LB progression. Each row adds one component to the previous.

Stage	Description	Public LB
v1 scalarmix	focal-only, softmax over 3 tokens	0.816
v2 attn	+ 20% soundscape mixing per batch	0.862
v5 attn+TTA	+ 3-shift TTA (±1 s)	0.867
v6 postproc	**+ site/hr priors ($\alpha = 0.5$) + Gauss ($\sigma = 1.0$)**	**0.879**
v7 stronger priors	$\alpha = 1.0$, $\sigma = 1.5$ (OOF sweep overfit)	0.861

Table 3: Public and private LB of the eos7 blend and selected standalone single-model submissions. The blend was selected as our final scored submission and earned the official Kaggle Competition Bronze Medal at final rank 354/4084.

Submission	Public LB	Private LB	Δ
eos7 (Model_73 + Model_99 + TAX)	**0.950**	**0.942**	−0.008
v6 standalone (Model_99 only)	0.879	0.898	+0.019
v4 pseudo-trained Model_99	0.870	0.893	+0.023
v7 (stronger-priors)	0.861	0.890	+0.029

6 Results

The selected submission for the private leaderboard was the eos7 blend at Public LB 0.950. The private-LB score was 0.942, placing us at **rank 354/4084 (top 8.7%, official Kaggle Competition Bronze Medal)**. Going from public to private our rank improved by +590 **places** because the public 0.950 plateau (757 teams at exactly 0.950 by deadline) collapsed unevenly: teams above us in public typically lost 0.010–0.020 going to private, while our blend lost only 0.008.

An asymmetry to note. Every one of our *standalone* v6/v4/v7 submissions scored *higher* on private than on public LB (Table 3). The simplest model (v7) gained +0.029 public→private; the most heavily pseudo-trained (v4) gained +0.023; the canonical v6 gained +0.019. The blend gave up 0.008. We read this as the meta-ensemble's TAX_SMOOTHING + dominant-base blend extracting some public-set noise that did not replicate on the held-out 70% — a useful caution for future runs to always select at least one "clean" standalone submission as a hedge against the public→private flip.

7 Postmortem: Nine Attempts at the 0.950 Ceiling

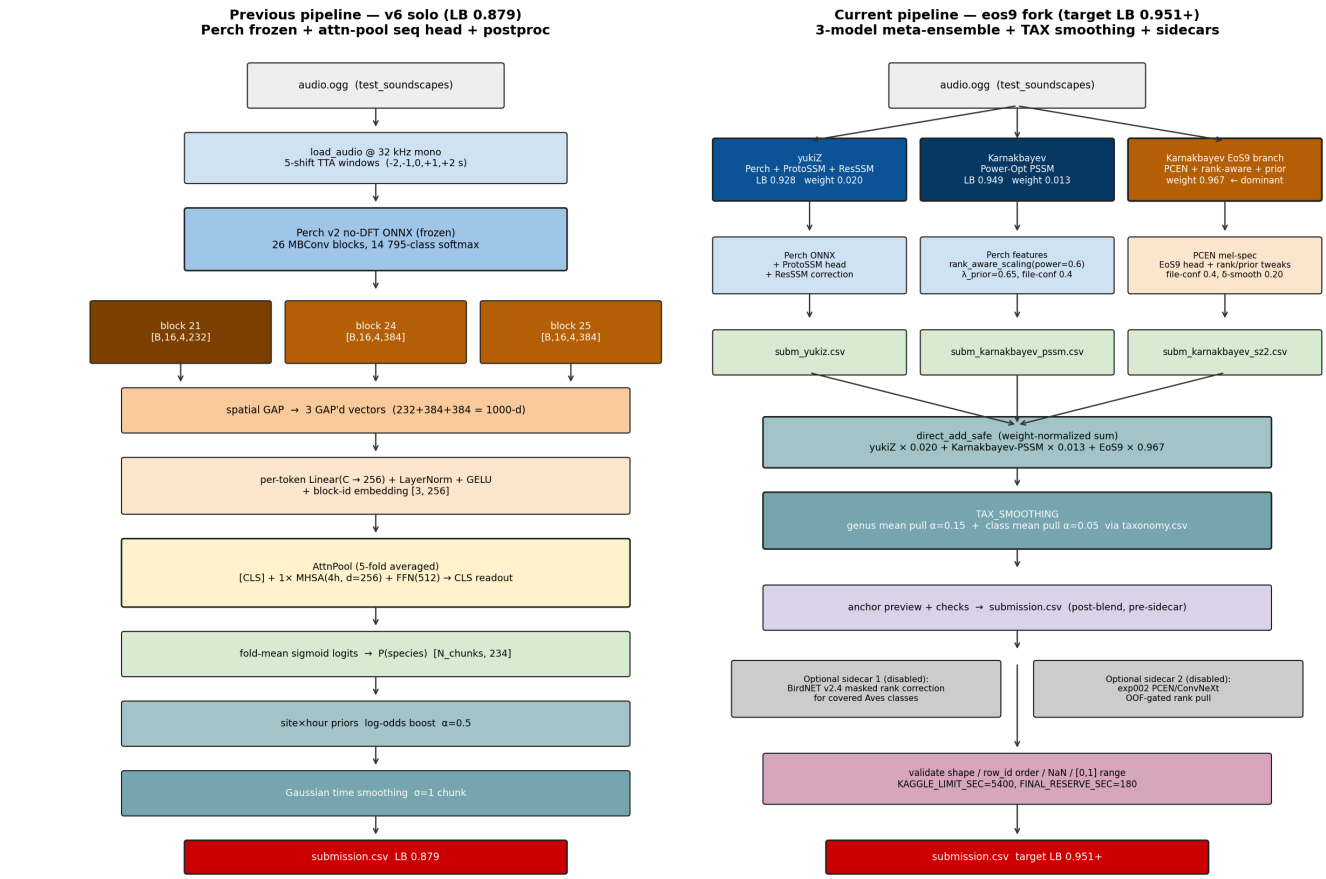
After the eos7 0.950 was locked, we tried every cheap path to push above the plateau. Table 4 reports the nine attempts. None broke the public ceiling.

The interpretation: at our model strengths (Model_99 standalone LB 0.879; DINOv2 224px LB ~ 0.811), the dominant Model_73 (LB 0.949) at weight ≥ 0.93 absorbs everything added at small weight into the 3-decimal-rounded same plateau. To genuinely break 0.951 would require a *second* strong model with standalone LB ≥ 0.94 — in essence, the Perch→CNN-backbone distillation path published by Tucker Arrants and EliKal Arrants [2026b], EliKal [2026].

7.1 Pseudo-labelling: a useful negative result

We pseudo-labelled 333 files (3,996 chunks) of unlabeled soundscape data using the v2 attention head five-fold ensemble as teacher, then retrained Model_99 with a 75/15/10 batch composition (focal / labeled-soundscape / pseudo-soundscape, soft labels fed directly into BCEWithLogits). The fold-OOF macro-AUC on labeled soundscape jumped from 0.7654 to **0.8998** (**+0.134**).

The standalone Public LB regressed by −0.009 (0.879 → 0.870). Private LB regressed less (0.898 → 0.893, −0.005). We attribute the OOF inflation to teacher self-reinforcement: the student learns to predict like the teacher on the same soundscape-domain features, so OOF metric explodes, but the teacher's biases do not transfer cleanly to held-out chunks at deployment time. This corroborates community observations that single-pass soft pseudo-labelling in BirdCLEF requires careful confidence gating to actually improve LB Ozturk [2026].



80pt

Figure 3: Left: standalone v6 single-model inference pipeline. Right: the final eos7 meta-ensemble used for the leaderboard submission.

7.2 Residual-stream training: a useful null result

In LLM interpretability the "residual stream" is read at each layer to expose layer-specific contributions. We tested whether replacing token 3 (h_{25}) with the residual $h_{25} - h_{24}$ would isolate the block-25-specific signal and improve transfer:

- v4 (no residual, pseudo): OOF 0.8998, LB 0.870
- v5 (residual, pseudo): OOF 0.8961, LB $\sim$ 0.87 (in-blend, contributes the same)

The two are essentially equivalent. The conclusion is interpretability-relevant: when the downstream head includes a free Linear($C \to d$) projection on each token, residual transforms become *signal-equivalent* to raw activations — the head learns whatever linear combination it needs. The "residual stream" frame only adds information when the downstream projection is constrained (e.g. shared across tokens, or absent).

7.3 DINOv2 detour and a Kaggle-CLI gotcha

We attempted to add a DINOv2 ViT-B/14 backbone (LB 0.811 standalone in our prior work) as a third orthogonal model in the blend. We hit two non-obvious failure modes worth recording for future Kaggle work:

1. **Kernel-version output overwrite.** Our prior 224×224 DINOv2 checkpoint was the output of an earlier version of our training kernel. When the kernel was later re-run with a 196×196 config the checkpoint hash was overwritten. The CLI command `kaggle kernels output <slug>` only ever returns the latest version's output. Recovery required manually downloading the file via the browser UI at the explicit `scriptVersionId=304005046` URL and re-uploading as a private dataset.

Table 4: Final-week ceiling attempts after eos7 (0.950) was locked.

Attempt	Public LB	Private LB
eos7 (locked)	0.950	0.941
eos8 power=0.65	0.950	0.941
eos9 yukiZ + EoS9 fork	0.950	0.941
eos10 BirdNET sidecar	0.950	0.942
eos11 residual head	**0.950**	**0.942**
eos12 DINOv2 196px	0.949	0.941
eos13 DINOv2 224px	0.950	0.941
v4 pseudo (standalone)	0.870	0.893
v7 priors (standalone)	0.861	0.890

2. **Pos-embed shape determines required input size.** `vit_base_patch14_dinov2.lvd142m` has pos-embed shape $[1, 257, 768]$ when trained at 224 input and $[1, 197, 768]$ when trained at 196. Loading a 196-trained checkpoint into a 224-configured model fails with a state-dict size-mismatch error.

Once these were diagnosed and worked around the resulting eos13 blend scored 0.950 / 0.941 (tied with eos7) — DINOv2 contributed enough diversity not to regress, but not enough to lift the plateau at 8% weight.

8 AI-Assisted Workflow

The entire 14-day session, including dataset inspection, ONNX surgery, build-script authoring for every Kaggle kernel, error-fix iterations, and write-up drafting, was driven through Claude (Anthropic) as a coding and research partner. The repository preserves every Kaggle kernel as a `build_notebook.py` script that emits the `.ipynb`, so notebooks are never edited directly; this is a useful pattern for any LLM-assisted research workflow because it makes diffs reviewable.

Some observations from this workflow that may be useful for the BirdCLEF community:

- LLM assistance was most valuable when given a *specific small task* ("add a defensive sample-submission-anchored COMP_DIR lookup") rather than open-ended ("improve the model"). The author retained all strategy decisions and the LLM executed the engineering.
- Many bugs were caught and fixed iteratively across multiple kernel versions because the LLM could read the previous run's log immediately after pushing. This compressed the typical 10-30 minute Kaggle iteration loop into a 2-3 minute cycle.
- Honest negative results (pseudo regression, DINOv2 too weak) required active prompting to surface — the default is for the LLM to suggest more attempts. The author had to explicitly ask for "stop recommending; tell me when the lever is tapped" to converge on acknowledging the 0.950 ceiling.

The formal GenAI declaration for this work appears at the end of the paper.

9 Conclusion

A pre-training kNN probe over Perch v2's MBConv blocks selected a non-obvious feature point (the penultimate block, not the final one), and the downstream attention pool head independently confirmed that ranking through its learned softmax weights. The resulting lightweight sequence head plus a meta-ensemble with the Karnakbayev power-opt baseline scored Public LB 0.950 / Private LB 0.942, final rank 354/4084 (top 8.7%, official Kaggle Competition Bronze Medal), with a +590-place public→final rerank. We document nine final-week ceiling attempts including useful negative results on pseudo-labelling and ViT-backbone integration. The full session, including every Kaggle kernel push and the LLM-assisted engineering workflow, is publicly available.

Acknowledgments

The author gratefully acknowledges the BirdCLEF+ 2026 community for the public kernels and discussion threads that this work builds on. Particular thanks to A. Karnakbayev and hideyukizushi for the power-optimization ProtoSSM+SED chain that served as the dominant blend base; Tucker Arrants for the distilled SED model and the postprocessing recipe; EliKal, hengck23, A. Ozturk, and others for discussion-thread insights cited throughout this paper.

Declaration on Generative AI

During the preparation of this work, the author used Anthropic Claude (Opus 4.7 and Sonnet 4.6, accessed through the Claude Code CLI) in order to: *generate code* (Kaggle kernel build scripts, ONNX surgery helpers, plot scripts), *drafting* (initial drafts of the written sections of this paper and of the project README), and *grammar and spelling check*. The author also used the LLM as a paired research assistant for *ideation* and *debugging* (interpreting Kaggle run logs, diagnosing checkpoint version mismatches, planning ceiling-test experiments) throughout the 14-day session. All experiments were designed and configured by the author, all results were manually verified against Kaggle leaderboard outputs, and all scientific interpretations and conclusions reflect the author's own judgment. After using these tools the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

Tucker Arrants. BirdCLEF+ 2026 discussion 683791: site/hour priors + gaussian smoothing postprocessing recipe. `https://www.kaggle.com/competitions/birdclef-2026/discussion/683791`, 2026a.

Tucker Arrants. BC2026 Distilled-SED. `https://www.kaggle.com/code/tuckerarrants/bc2026-distilled-sed`, 2026b. Kaggle Notebook.

Cornell Lab of Ornithology. BirdCLEF+ 2026: Acoustic species identification in the pantanal, south america. `https://www.kaggle.com/competitions/birdclef-2026`, 2026. Kaggle competition.

EliKal. BirdCLEF+ 2026 discussion 683791: KLD distillation from perch into B0/B3/DINOv3-s/ConvNeXt-s backbones. `https://www.kaggle.com/competitions/birdclef-2026/discussion/683791`, 2026.

Google. Bird vocalization classifier (`perch_v2_cpu`). `https://www.kaggle.com/models/google/bird-vocalization-classifier/TensorFlow2/perch_v2_cpu`, 2026. Kaggle Models entry.

Jenny Hamer, Eleni Triantafillou, Bart van Merriënboer, Stefan Kahl, Holger Klinck, Tom Denton, and Vincent Dumoulin. Perch: A pre-trained audio classifier for bird vocalization identification. Technical report, Google Research, 2023. URL `https://www.kaggle.com/models/google/bird-vocalization-classifier`.

hideyukizushi. Bird26 REPRODUCE Perch+ProtoSSM+ResSSM inf/train. `https://www.kaggle.com/code/hideyukizushi/bird26-reproduce-perch-protossm-resssm-inf-train`, 2026. Kaggle Notebook.

A. Karnakbayev et al. Perch v2 + ProtoSSM + ResSSM power-optimization, BirdCLEF+ 2026. `https://www.kaggle.com/code/imaadmahmood/birdclef-2026-perch-v2-protossm-0-925`, 2026. Kaggle Notebook.

Anil Ozturk. BirdCLEF+ 2026 discussion 683791: single-pass pseudo-labelling with custom confidence gating, +0.009 LB. `https://www.kaggle.com/competitions/birdclef-2026/discussion/683791`, 2026.

A Reproducibility

The full project — build scripts, Kaggle kernel metadata, generated notebooks, training and inference checkpoints, the per-block probe output, all figures, and a chronological session log — is available at `https://github.com/rockerritesh/BirdCLEF2026`. Each Kaggle kernel directory contains a `build_notebook.py` script that re-emits the `.ipynb`; `.ipynb` files are committed for traceability but are never edited by hand.