BirdCLEF+ 2026

Professional agenda and execution roadmap for a strong Kaggle campaign

Focus: EDA, validation, model testing, hyperparameter tuning, ensembling, stacking, expert learning, and final-push operations

Eric Benhamou

Agenda

Phase 1

- Problem framing and data inventory
- EDA, metadata audit, and leakage checks

Phase 2

- Validation design and offline metric stack
- Fast replay protocol and stability tests

Phase 3

- Baseline ladder and model-family testing
- CPU-aware inference optimization

Phase 4

- Hyperparameter tuning
- Sampling, augmentation, loss, and training controls

Phase 5

- Pseudo labels, distillation, ensembling, stacking
- Error analysis and expert learning loop

Phase 6

- Weekly cadence and final two-week push
- Submission hardening and decision rules

Competition snapshot

Task

Identify species from passive acoustic recordings collected in the Pantanal.

Output

Per-species probabilities for each 5-second audio window.

Metric

Macro-averaged ROC-AUC, skipping classes with no true positives.

Target space

234 species columns in the submission file.

Submission runtime

CPU notebook only, with a hard 90-minute runtime constraint.

Extra leverage

Some train_soundscapes are labeled by expert annotators this year.

What this means for strategy

This is not a pure “train a bigger model” competition. Winning requires a strong validation design, domain-shift handling, and a final model stack that is both accurate and fast enough for CPU inference.

Why BirdCLEF 2026 is difficult

- Training clips and deployment soundscapes do not follow the same distribution.
- Multiple taxa can overlap in the same window, often under heavy background noise.
- The label space is long-tailed: rare classes matter, but head classes dominate volume.
- Submission quality is constrained by CPU-only inference and notebook runtime.
- The public leaderboard only reflects a subset of the test set, so validation must lead.

Shift
Clean train clips → messy field soundscapes

Overlap
Birds + frogs + insects + noise

Latency
Accuracy must survive CPU deployment

Winning principles before any serious modeling

1. Validation first

Decide the fold logic before tuning models; otherwise the leaderboard will mislead you.

2. Build a baseline ladder

Move from sanity checks to strong baselines to advanced ideas; do not jump directly to a giant ensemble.

3. Keep inference in loop

Any promising idea must remain compatible with the final CPU notebook.

4. Exploit unlabeled audio

BirdCLEF winners repeatedly gain from pseudo labels, soundscape reuse, and distillation.

5. Optimize diversity

Ensembling works when models differ in backbone, features, labels, or training recipe.

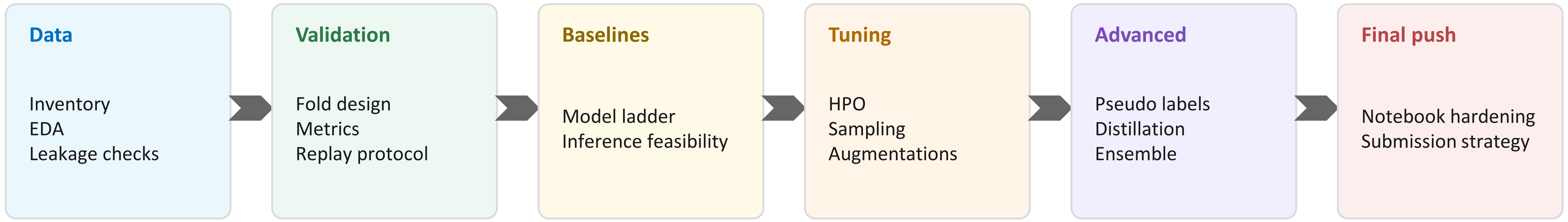
6. Track every experiment

Model quality without lineage is not usable in the final push.

Working rule

Every branch should answer one question only: better validation score? more stable across folds? faster inference? more ensemble diversity? If the answer is unclear, prune it.

Roadmap at a glance



Recommended operating model

- Weeks 1–2: understand data and lock validation.
- Weeks 3–4: establish the strongest single-model baseline.
- Weeks 5–6: systematic tuning and soundscape exploitation.
- Final phase: ensemble only validated, CPU-feasible components.

01

Phase 1 — Problem framing and data inventory

Understand the competition before touching heavy modeling.

Data inventory: what to inspect on day 1

train_audio

Primary supervised source; inspect durations, metadata coverage, label distribution, and recording quality.

train_soundscapes

Closest domain to the test set; use for EDA, validation design, and semi-supervised ideas.

test_soundscapes

Infer deployment conditions from filenames, duration pattern, time structure, and row mapping.

metadata tables

Site, date, taxonomy, quality, or provenance fields often determine the right group split.

sample_submission

Confirms the real prediction unit: species probabilities per 5-second row.

discussion + starter code

Look for organizer clarifications, exact constraints, and baseline assumptions.

Deliverable

Produce a one-page data memo: file inventory, schema, label counts, suspected leakage variables, and an initial risk list.

Label space and submission target

- There is one prediction column per target species.
- Each submission row corresponds to a 5-second window.
- The model must output probabilities, not hard decisions.
- Because the competition metric is threshold-free, ranking quality matters more than manual threshold tuning during training.
- Rare classes still matter: the validation view must include class-level behavior, not only a single global score.

Prediction unit

row_id = soundscape segment
Target = probability for each species

Training implication

Treat the task as multilabel ranking over short audio windows.

Evaluation implication

Optimize calibrated relative ordering of positives vs negatives for every class.

Domain shift map: where the competition is won or lost

Training clips

- Usually cleaner
- Often centered on the target vocalization
- Stronger labels
- Less overlapping fauna
- Metadata may reflect provenance more than deployment context

Test soundscapes

- Continuous field recordings
- Multiple species per window
- Wind, rain, insects, humans, machinery
- Different acoustic conditions across place and time
- Much closer to real deployment

Implication 1

Validation must simulate soundscape deployment.

Implication 2

Pseudo labels are structurally valuable.

Implication 3

Fast students matter as much as big teachers.

EDA checklist: what to learn before training

Listen to samples

Head classes, rare classes, false labels, background regimes

Browse spectrograms

Call shape, silence ratio, harmonics, clipping, non-target events

Measure class counts

Long tail, species coverage per fold, rare-class minimums

Inspect durations

Clip-length bias and usable crop windows

Audit metadata

Site, time, habitat, provenance, and any group-split keys

Check co-occurrence

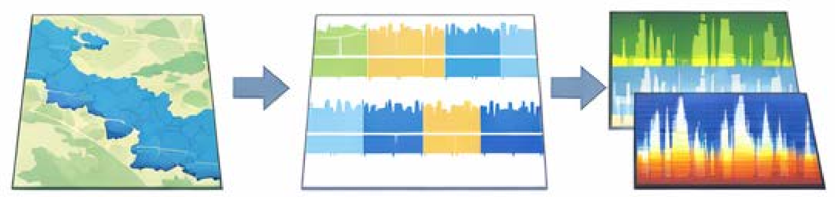
Which species or taxa tend to appear together in soundscapes

Rule of thumb

Do not advance to hyperparameter tuning until the data memo explains class imbalance, domain shift, and the first leakage hypothesis.

Acoustic inspection views to standardize

- Waveform: clipping, silence, amplitude drift, and obvious corruption.
- Log-mel spectrogram: the main input view for CNN-style baselines.
- PCEN or denoised view: useful when background energy dominates.
- Per-band energy / SNR proxy: quickly detect low-information clips.
- Windowed browsing: inspect how a 1-minute soundscape decomposes into 5-second targets.



Standard view A

5-second windows for training / inference parity

Standard view B

Longer windows for error analysis and context studies

Metadata audit: variables that often matter more than expected

Potential group keys

site_id
location
recordist
date block
habitat
device or provenance

Potential leakage clues

duplicate files
near-duplicate clips
shared soundscape ancestry
recording-session overlap

Useful feature enrichments

time of day
month / season
broad habitat bucket
quality score bucket

Risk

A random split can look much better than true deployment.

- Map every metadata field to one of three roles: feature, split key, or leakage risk.
- Quantify missingness and cardinality before using any field in modeling.
- Build derived groups early so all folds and experiments reuse the same grouping logic.

Output of this step

A frozen metadata table, versioned and ready to drive fold construction, class analysis, and downstream experiment joins.

Leakage and QA guardrails

Checks to automate

Hash duplicates, near-duplicates, label-map consistency, bad audio, corrupt metadata joins, and fold leakage across related records.

Checks to review manually

Suspicious top validation clips, impossible co-occurrences, mislabeled negatives, and species with extremely atypical spectrograms.

Guardrail 1

Fold assignment must be generated once and reused everywhere.

Guardrail 2

Every feature table must carry a sample identifier and a provenance column.

Guardrail 3

OOF predictions, not public-LB scores, decide whether an idea survives.

Exit criterion for Phase 1

You can explain the dataset, defend the fold logic, and trust the experiment tables enough to start systematic validation.

02

Phase 2 — Validation design

Build the offline score you will actually trust.

Why validation decides this competition

- The public leaderboard uses only part of the test data, so it is a noisy decision signal.
- BirdCLEF has strong domain shift; a high score on the wrong split can be meaningless.
- OOF predictions are needed later for ensembling, stacking, threshold studies, and error analysis.
- A model with slightly lower mean but lower fold variance is often a better final candidate.

Decision hierarchy

1) Mean OOF score
2) Fold stability
3) Subgroup robustness
4) Runtime
5) Public LB only as a weak cross-check

Do not optimize to

Temporary public-LB bumps from weak CV, accidental leakage, or runtime-heavy notebooks.

Do optimize to

A reproducible offline pipeline that predicts private-LB performance and supports fast iteration.

Candidate cross-validation schemes

Random multilabel stratification

Fast and often optimistic. Good only as a sanity baseline.

Group split by site / session

Best when metadata identifies shared acoustic conditions.

Time-aware split

Useful if seasonality or acquisition waves create shift.

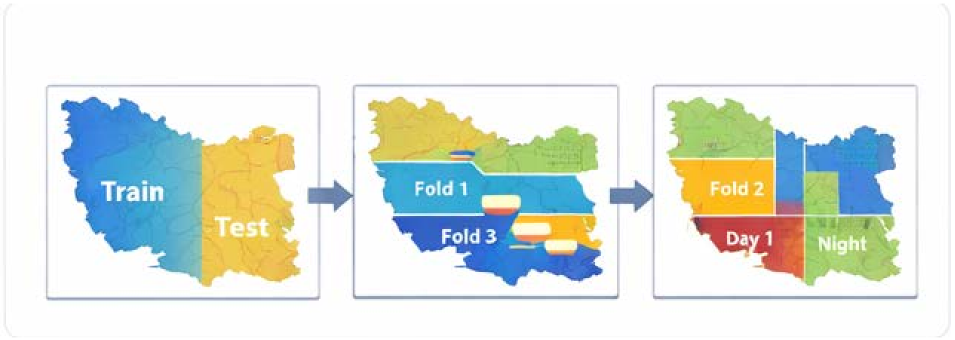
Hybrid grouped + stratified

Usually the strongest practical option for BirdCLEF-like data.

Recommendation

Test 2–3 candidate split families early using a small replay protocol. The winner is the split that best differentiates robust ideas from fragile ones, not necessarily the one with the highest raw score.

Recommended fold design for BirdCLEF 2026



- Use 5 grouped folds as the default backbone for all serious experiments.
- Prefer grouping keys tied to soundscape context: site, session, day block, or other recording clusters.
- Within that constraint, rebalance fold class counts so rare classes appear often enough to evaluate.
- Keep one shadow split for “stress testing” domain shift, even if it is not the official tuning split.

Primary split

Grouped 5-fold CV

Shadow split

Harder stress split

Goal

Generalization > convenience

Offline metric package

Primary

Macro ROC-AUC on OOF predictions
Use the competition-style class handling as closely as possible.

Secondary

Rare-class PR-AUC or per-class ranking tables
Useful when the head classes dominate interpretation.

Operational

Runtime, memory, and notebook feasibility
Treat these as first-class metrics.

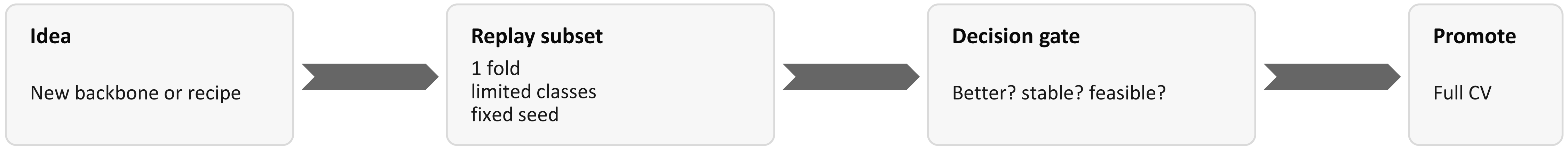
Subgroup view

Track score slices by fold, time of day, site group, rare vs frequent class bucket, and overlap-heavy windows. The average alone is not enough.

Metric rule

Promote only ideas that improve the primary score without creating obvious regressions in subgroup robustness or runtime.

Fast replay protocol for cheap idea triage



- Replay protocols cut dead-end ideas before expensive full-fold runs.
- Fix the subset once so comparisons remain fair.
- Keep a strict promotion threshold to protect compute budget.

Practical target

The best teams usually operate multiple experimental speeds: replay, medium CV, and full final validation.

Stability tests that prevent leaderboard surprises

Fold variance

Mean without dispersion is weak evidence.

Seed variance

Check whether gains survive random initialization.

Subgroup robustness

Stress day/night, site bucket, or overlap-heavy slices.

Public-LB relation

Use only as a loose directional cross-check.

Runtime margin

Leave safety headroom below the notebook cap.

Reproducibility

Same config hash should reproduce the same metrics.

Validation dashboard and go/no-go rules

Model family / config	**OOF mean** primary metric	**Fold std** stability	**Runtime** CPU feasibility	**Diversity** ensemble value	**Decision** promote / prune
CNN baseline	**reference**	**low**	**safe**	**medium**	**promote**
Pretrained bird model	**higher?**	**medium**	**watch**	**high**	**test more**
Context model	**unclear**	**high**	**risky**	**high**	**replay first**
Huge slow model	**maybe**	**medium**	**unsafe**	**low**	**prune**

Rule
No model reaches the final ensemble unless it is measurable in this dashboard.

03

Phase 3 — Baselines and model testing

Create a strong ladder from simple to advanced.

Baseline ladder

L0	Sanity baseline	Class priors, constant outputs, submission plumbing
L1	Classical audio features	MFCC / mel statistics + linear or tree models
L2	CNN spectrogram baseline	A reliable single-model deep baseline
L3	Pretrained audio / bird model	Transfer learning from stronger audio priors
L4	Context + SSL	Use soundscapes, pseudo labels, and longer context
L5	Validated ensemble	Only after OOF and runtime evidence

Classical feature baseline: useful even if it will not win

• Compute cheap descriptors: MFCC statistics, log-mel summaries, PCEN summaries, energy and silence features.
• Train one-vs-rest linear models or lightweight tree models as a sanity reference.
• Purpose: validate labels, metadata joins, and rare-class learnability before deep training.

Why keep it

A classical baseline often catches data bugs and occasionally provides diversity for later blends.

Why not overinvest

The leaderboard is usually decided by stronger deep audio representations plus soundscape exploitation.

Success criterion

The baseline should be reproducible, fast, and clearly worse than the best deep baseline — otherwise something may be wrong in the deep pipeline.

CNN spectrogram baseline: the first serious anchor

• Convert audio windows to log-mel or PCEN spectrograms.
• Use a robust image backbone such as EfficientNet or ConvNeXt as the first strong baseline.
• Keep the recipe simple: 5-second crops, BCE-style loss, moderate augmentation, and grouped CV.

Default recipe

224×224-ish input • mixed precision for training • OOF storage from day one

Immediate questions

log-mel vs PCEN?
5s only or context?
class weighting?
which sampler?

Why this matters

This model becomes the benchmark every later idea must beat.

Pretrained audio and bird models

• Prior BirdCLEF solutions repeatedly benefit from transfer learning and prior-domain audio representations.
• Test a compact set of pretrained families rather than a long, noisy model zoo.
• Judge them on three axes together: OOF score, runtime, and complementarity vs the CNN baseline.

Option A

Bird-specialized teacher

Option B

General audio model

Option C

Small CPU-ready student

Context and sequence models

• A single 5-second window can be ambiguous; neighboring windows often provide useful context.
• A practical design is encoder + temporal pooling / attention over adjacent segments.
• Context models can improve robustness, but they also increase training and inference complexity.

Test conservatively

First prove value in offline CV with a narrow context window.

Deployment question

Can context be approximated in the final CPU notebook without breaking runtime?

Decision rule

Keep context models only if they are both stronger and operationally manageable.

Inference optimization under CPU-only submission

- Notebook submissions effectively need a CPU-only pipeline with comfortable runtime headroom.
- Prefer backbones that convert cleanly to ONNX / OpenVINO or another CPU-friendly format.
- Batch window processing, feature caching, and compact students often matter as much as raw architecture choice.

Optimization levers

smaller backbones • compiled inference • fewer feature recomputations • efficient batching

What to measure

end-to-end wall time, peak memory, and runtime safety margin

Hard rule

A model that cannot survive notebook deployment is a teacher candidate, not a final submission candidate.

Model test matrix

Family	Expected score	Speed	Diversity	Best use
Classical features	Low–Med	High	Medium	sanity / possible blend
CNN spectrogram	High	Med–High	Base	main anchor
Pretrained bird model	High	Medium	High	teacher or final ensemble
Context model	Unknown	Low–Med	High	only if CV justifies it

Interpretation

The matrix is qualitative by design: it prevents overfitting to one metric too early.

04

Phase 4 — Training optimization and hyperparameter tuning

Tune systematically, not emotionally.

Hyperparameter tuning framework

Stage A

Lock data and CV; tune only the strongest baseline.

Stage B

Tune one axis at a time: sampler, loss, augmentation, optimizer.

Stage C

Run interaction checks only for the most promising settings.

Stage D

Freeze finalists.

- Use a replay split for broad search; escalate only the best candidates to full CV.
- Never retune multiple variables blindly after every leaderboard movement.
- Keep a compact search log with config hash, question asked, and answer learned.

Sampling and imbalance control

• BirdCLEF label distributions are usually long-tailed; sampling policy can move the score materially.
• Compare weighted samplers, class-aware batches, head-class caps, and rare-class oversampling.
• Watch for calibration damage: aggressive oversampling can make deployment probabilities unstable.

Practical rule

Use sampling to make rare classes visible — not to create a fake training distribution.

What to monitor

OOF rare-class score, global score, and probability calibration together.

Suggested policies

weighted sampler
class-aware batches
head-class caps
rare-class oversampling

Augmentation plan

Safe

time shift • gain • mild noise

Usually useful

SpecAugment • background mix • random crop

Case-by-case

bandpass filters • tempo perturbation • denoise transforms

Use with care

heavy warping or unrealistic acoustic distortions

Selection principle

Augmentations should improve robustness to real field conditions — wind, background fauna, level shifts, truncation — without changing class semantics so much that the offline metric becomes noisy.

Loss functions and training objectives

BCE baseline

Stable starting point for multilabel ranking.

Focal / asymmetric

Useful when negatives dominate and rare positives matter.

Label smoothing

Can help noisy labels; test carefully.

Class weighting

Tied to sampler choice and calibration.

• Because the competition metric is ranking-based, the best loss may be the one that yields the strongest OOF ordering rather than the nicest raw probabilities.
• Loss, sampler, and augmentation interact. Tune them as a small package, not in isolation forever.

Segment length, hop size, and spectrogram resolution grid

Segment length

Default: 5 s
Try shorter for brief calls
Try longer for context studies

Hop / overlap

Match submission unit first
Then test overlap only if offline gains are clear

Mel resolution

128–256 bins is the practical search range for most baselines

Frequency range

Task-aware low/high cut depending on taxa mix

Search plan

Keep this grid small. The goal is not to maximize combinations; it is to identify one or two input settings that are consistently strong enough to support the rest of the tuning stack.

Baseline
5 s • default hop • 128/256 mel

Context variant
8–10 s • context study only

Short-call variant
2.5–3 s • useful for brief vocalizations

Optimizers, schedules, and regularization

• Start from AdamW or another stable adaptive optimizer.
• Use cosine or one-cycle style schedules only after the baseline is healthy.
• Compare dropout, weight decay, EMA, and early stopping with strict logging.

Order of importance

1) good data split
2) good sampler / loss
3) architecture choice
4) optimizer refinements

Common mistake

Over-searching schedules before solving data shift and imbalance.

Recommendation

Tune schedule families only on the strongest 1–2 candidate baselines.

Experiment tracking and pruning

Track

config hash
fold assignment version
data version
seed
OOF metrics
runtime

Tag

baseline
replay
full CV
teacher
student
ensemble candidate

Decide

promote
park
kill
merge into ensemble pool

Pruning policy

If an experiment does not answer a clear question or cannot enter a later ensemble pool, stop it. The competition is short; memory and discipline beat volume.

Output

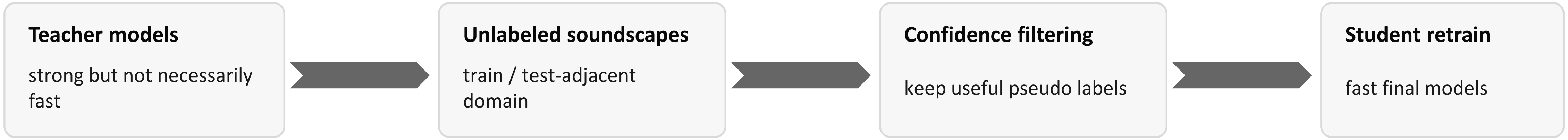
A compact leaderboard of your own experiments — more valuable than the public leaderboard.

05

Phase 5 — Advanced improvement

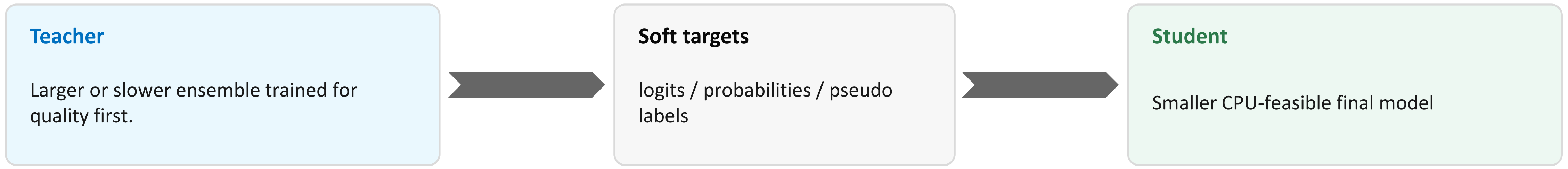
Exploit soundscapes, model diversity, and expert lessons.

Pseudo-labeling strategy



- This is one of the recurring themes in recent BirdCLEF writeups.
- Use pseudo labels conservatively at first: confidence filters, rare-class review, and a fixed ratio vs clean labels.
- OOF-based teacher selection is safer than public-LB-based teacher selection.

Distillation and compression



Teacher

Larger or slower ensemble trained for quality first.

Soft targets

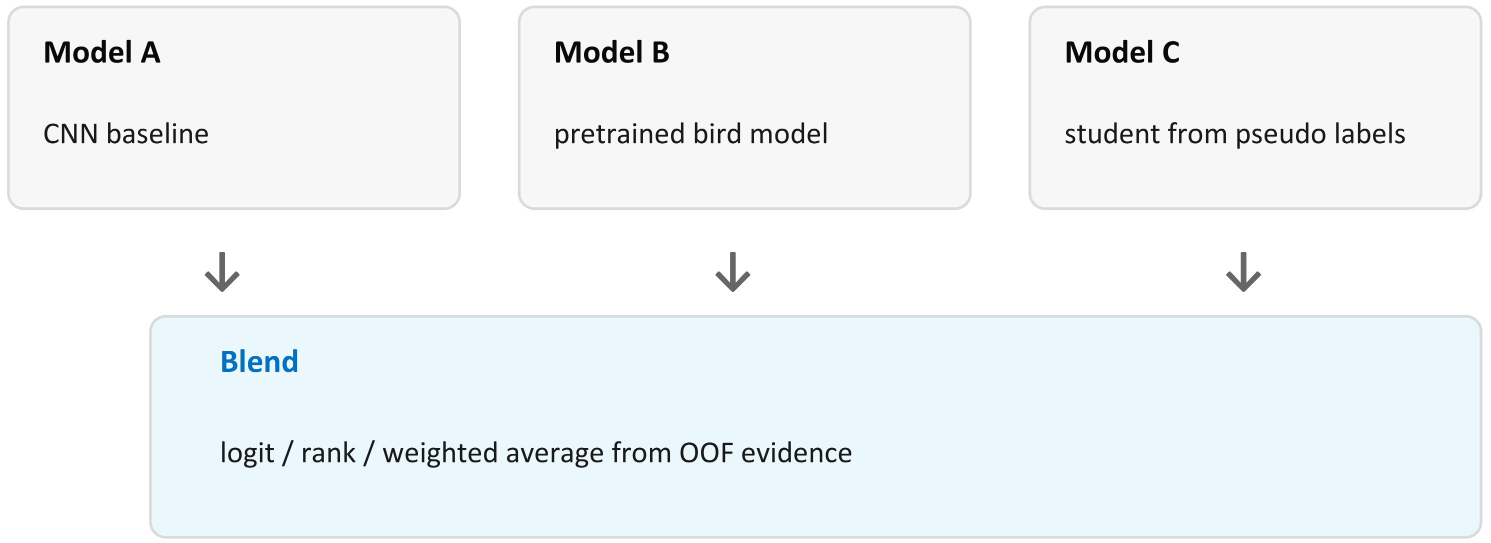
logits / probabilities / pseudo labels

Student

Smaller CPU-feasible final model

- Distillation converts model quality into deployable quality.
- Use it when the best teacher family is too slow or too memory-heavy for final submission.
- Measure whether the student preserves the teacher’s subgroup behavior, not just the global score.

Ensembling design


Model A
CNN baseline
Model B
pretrained bird model
Model C
student from pseudo labels
Blend
logit / rank / weighted average from OOF evidence

- Ensemble only models that are both strong and different.
- OOF correlation matters: three nearly identical models rarely beat two diverse ones.
- Keep a safe ensemble and a bold ensemble until the final notebook tests are complete.

Stacking and blending logic

Level-1 data

Out-of-fold predictions from the base models only.

Meta model

Simple blender first: linear / ridge / per-class weights.

Leakage rule

Never fit the stacker on predictions generated from the same training targets without OOF separation.

- Start with weighted blending before trying a more flexible stacker.
- The stacker earns its keep only if it beats simple blending on the same OOF frame.
- Keep the final notebook simple; a fragile stacker can fail under time pressure.

Calibration and post-processing

Calibration

Useful for interpretability, threshold studies, and pseudo-label quality control.

Temporal smoothing

Adjacent windows can regularize noisy frame-level outputs.

Class priors

May help if some classes are systematically over- or under-confident.

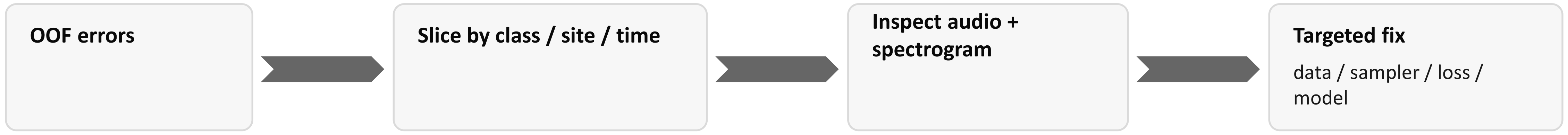
Practical view

Because the official metric is threshold-free, post-processing should be evaluated by OOF ranking improvements, not by how “nice” the probabilities look.

Use case

Especially useful for pseudo-label curation, student training, and final error analysis.

Error analysis loop


OOF errors
Slice by class / site / time
Inspect audio + spectrogram
Targeted fix
data / sampler / loss / model

- Error analysis is where the next worthwhile experiment should come from.
- Track recurring failure modes: rare-class miss, overlap confusion, background noise, habitat drift, and overconfident false positives.
- A competition branch without a diagnosed failure mode tends to waste time.

Learn from experts systematically

Discussions

Organizer clarifications, CV hints, runtime issues, hidden pitfalls.

Public notebooks

Replicate only the high-signal baselines worth understanding.

Winning writeups

Extract recurring ideas, not just one-off tricks.

Repos / working notes

Look for deployment, distillation, and data-engineering details.

- Recurring BirdCLEF themes across recent years: transfer learning, pseudo labels, distillation, and careful CPU inference engineering.
- Make a living checklist of ideas you have reproduced, rejected, or postponed.
- Do not cargo-cult a winning solution: revalidate it under your split and notebook budget.

06

Phase 6 — Operations and final push

Turn a research pipeline into a competition pipeline.

Weekly execution cadence

Week 1	Week 2	Weeks 3–4	Weeks 5–6	Final phase
Deliverables	**Deliverables**	**Deliverables**	**Deliverables**	**Deliverables**
Data memo EDA pack fold proposal	Locked CV strong baseline replay pipeline	Model family test CPU feasibility first OOF dashboard	Tuning pseudo labels student candidates	final ensemble notebook hardening submission strategy

Operating rhythm

One stable weekly review: what improved OOF, what failed, what enters the final ensemble pool, and what gets killed.

Final two-week push

• Freeze the validation logic; do not redesign CV late unless a critical bug appears.
• Retrain only the shortlisted models and produce clean OOF artifacts for blending.
• Dry-run the final inference notebook multiple times with timing logs and safety headroom.
• Keep two submission tracks: a conservative, proven blend and a higher-upside experimental blend.
• Document every final decision so the last submission day is operational, not emotional.

Checklist

OOF files frozen
weights frozen
runtime confirmed
seed / config archived

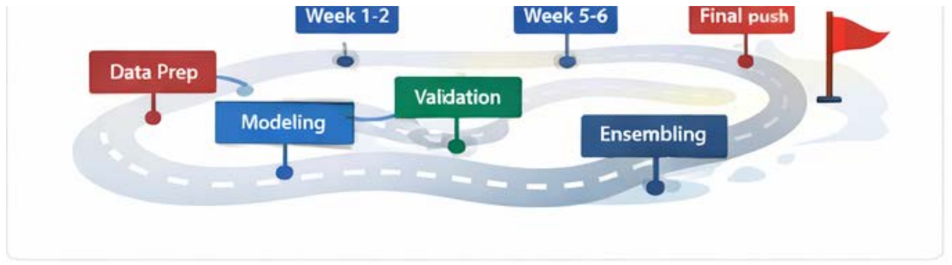
Risks

late overfitting
runtime regressions
missing artifacts
hidden leakage in stacker

Mitigation

safe fallback notebook
small final candidate set
full reproducibility

Closing recommendations and first 10-day plan



- Days 1–3: complete the data memo and leakage checklist.
- Days 4–5: lock grouped CV and build the replay protocol.
- Days 6–8: train the first serious CNN baseline and benchmark CPU feasibility.
- Days 9–10: test one pretrained family and define the initial tuning backlog.

Main advice

Protect validation.

Main advice

Exploit soundscapes.

Main advice

Keep CPU deployment central.

Bottom line

The best BirdCLEF strategy is a disciplined system: strong CV, fast baselines, targeted tuning, soundscape leverage, and only evidence-backed ensembles.