PhD Summer School on Uncertainty in Machine Learning

Predictive Uncertainty Challenge 2026

Don't just predict — predict responsibly.

Predictive Uncertainty Challenge 2026

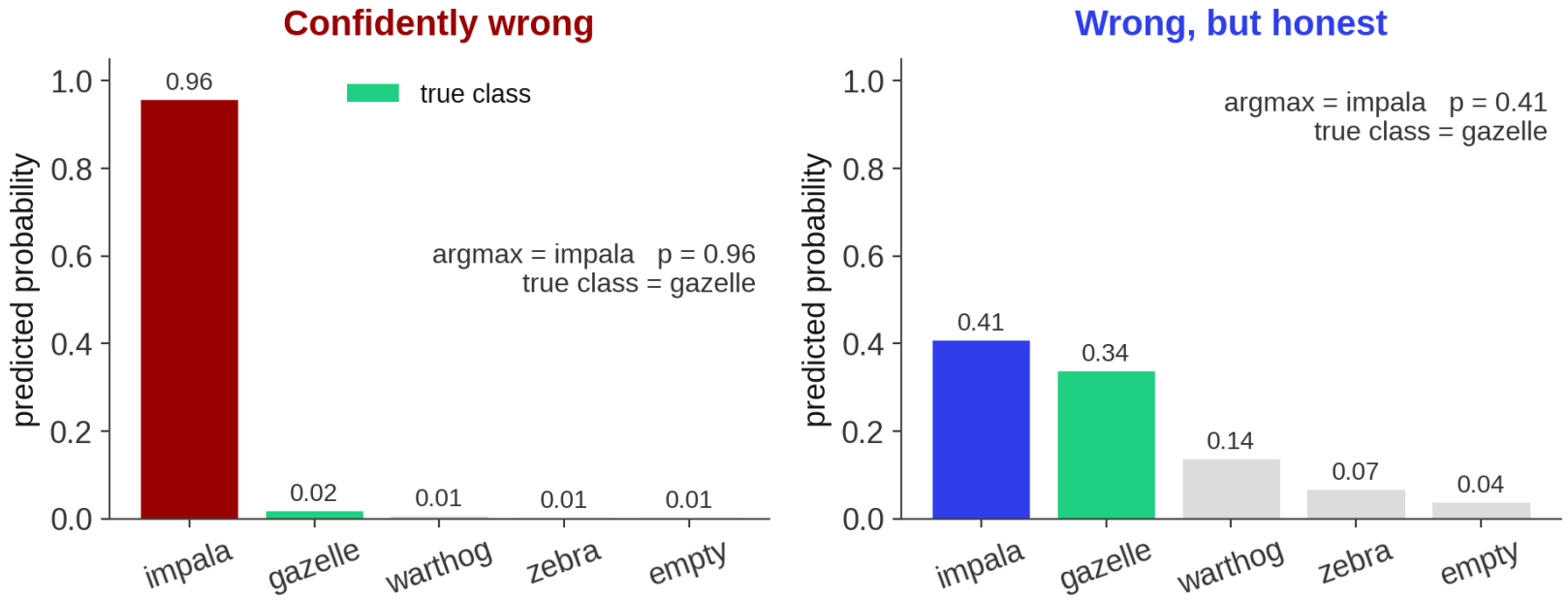
- Main goal of the challenge:
  - Classify wildlife species in camera-trap images — and say how sure you are
  - A model that is slightly less accurate but well calibrated is often far more useful than one that is wildly overconfident
  - Dataset: iWildCam (WILDS) — real motion-triggered camera traps, in-distribution and held-out locations
  - You are scored on four numbers — ECE, NLL, Brier and misclassification AUROC — none of which is plain accuracy

Photo credit: iWildCam challenge (github.com/visipedia/iwildcam_comp)

Why is this relevant?

A wrong prediction is a problem. A confidently wrong prediction is a bigger one.

Same accuracy. Very different usefulness.



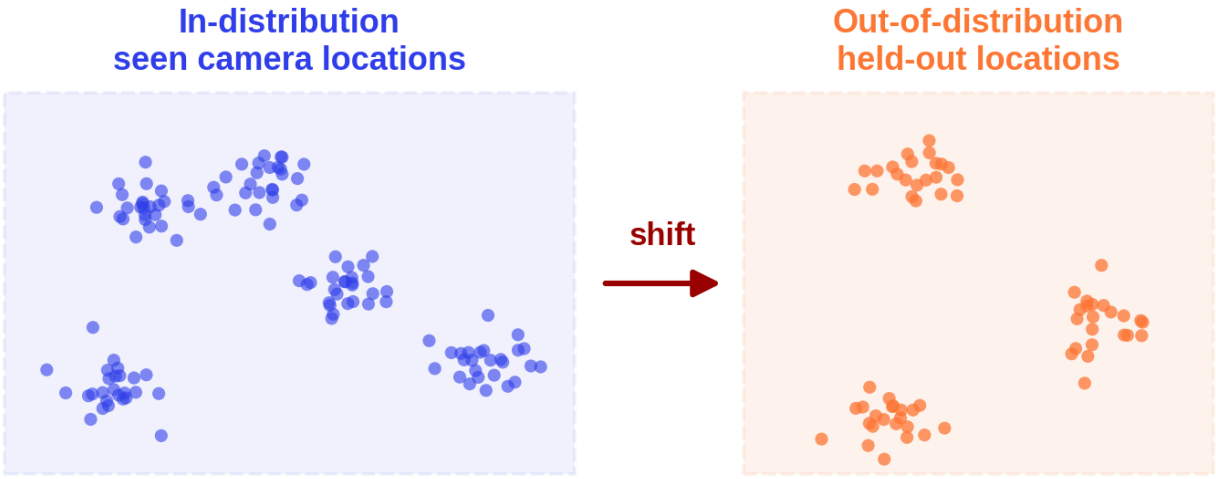
Downstream users act on the probability, not just the label: ecologists triage which images a human reviews, clinicians decide when to order another test, autonomous systems decide when to stop.

If the model knows when it does not know, the whole pipeline gets safer.

Why is this hard? Distribution shift

Camera traps move. Your test images come from locations the model has never seen.

New locations mean new backgrounds, lighting and species mixes —
the model must stay calibrated where it has never been



Animals may be centred, or a tiny blur in the corner; in daylight, or in infrared night mode; alone, or photobombed.

Accuracy drops on new locations — the question is whether your confidence drops with it.

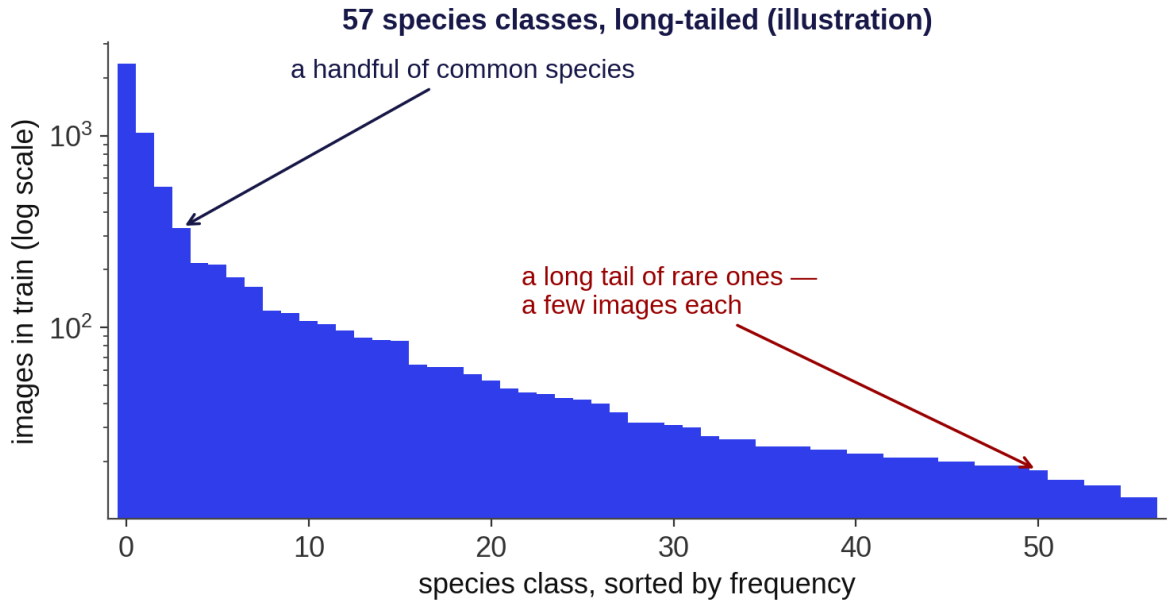
What are you supposed to do?

- Train a species classifier on the provided camera-trap images
- Make its probabilities trustworthy — not just numerically convenient
- Handle domain shift: new camera locations, lighting and species mixes
- We provide a starter codebase: dataloading, a pretrained CNN baseline with a swappable backbone, temperature scaling, a local evaluator and a submission script
- You should:
  - Train and evaluate the baseline, then improve on it
  - Pick a strategy for uncertainty: calibration, ensembles, MC dropout, loss re-weighting …
  - Submit probability predictions to the challenge server and climb the leaderboard
  - Do well on all four scored metrics — they are combined with a Borda count, so consistency beats one-trick ponies

ECE | **NLL** | **Brier** | **Misclass. AUROC**

Every submission is scored on all four — details in a moment.

Data – iWildCam camera traps



- Images from motion-triggered camera traps around the world

- 57 species classes remain after we drop the classes that never appear in the held-out locations — so K = 57, not the 182 of the WILDS release
- Long-tailed: a few species dominate, many are rare
- Some camera locations are shared with training (in-distribution), others are held out (out-of-distribution)

Data – what you get, and what we hold back

train – labels given

- Images + labels.csv (uid, y, domain)
- In-distribution camera locations only
- Subsampled per class from the WILDS train split

test_public – no labels

- Images only — predict probabilities for every uid
- Drives the live leaderboard during the week
- Mixed id / ood, roughly 60 % id and 40 % ood

val – labels given

- Images + labels.csv (uid, y, domain)
- Equal parts id (seen locations) and ood (held-out)
- Your local evaluator + temperature scaling run here
- Use the domain column for split-conditional analysis

test_private – no labels

- Images only — predict these too
- Never scored publicly during the challenge
- Used for the final ranking on Thursday
- The complement of test_public in the held-out pool

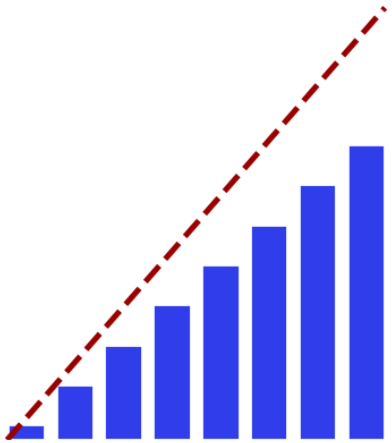
One submission covers both test splits.

student.predict writes a single CSV with one row per uid across test_public and test_private. The server scores the public rows for the leaderboard and keeps the private rows for the final result — so you cannot tune against the number that decides the winner.

How you are scored

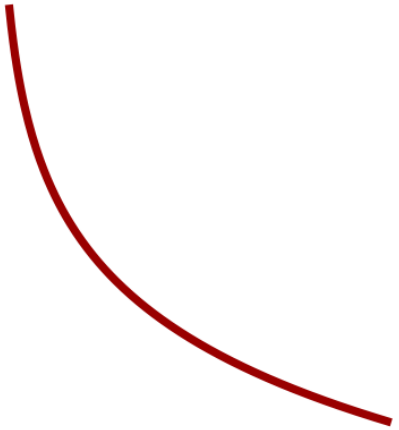
Four numbers, each asking a different question about your probabilities

ECE



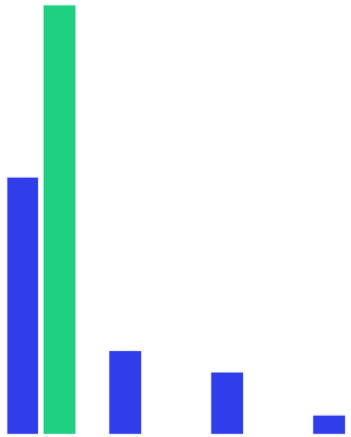
does confidence match accuracy?

NLL



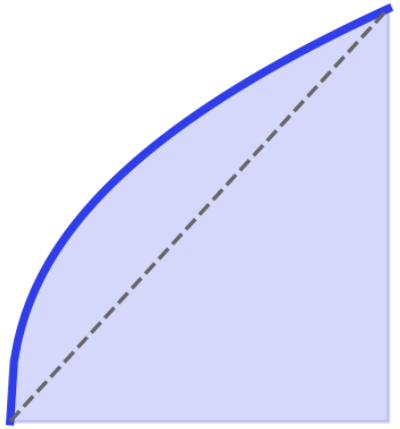
how much mass on the true class?

Brier



distance to the whole one-hot vector

Misclass. AUROC



is the model more confident when right?

Every one of them reads the probabilities themselves, not just the arg max — plain accuracy is shown on the server but is not part of your score.

Two are proper scoring rules (NLL, Brier): they reward being both sharp and honest. One measures calibration only (ECE). One measures ranking only (AUROC).

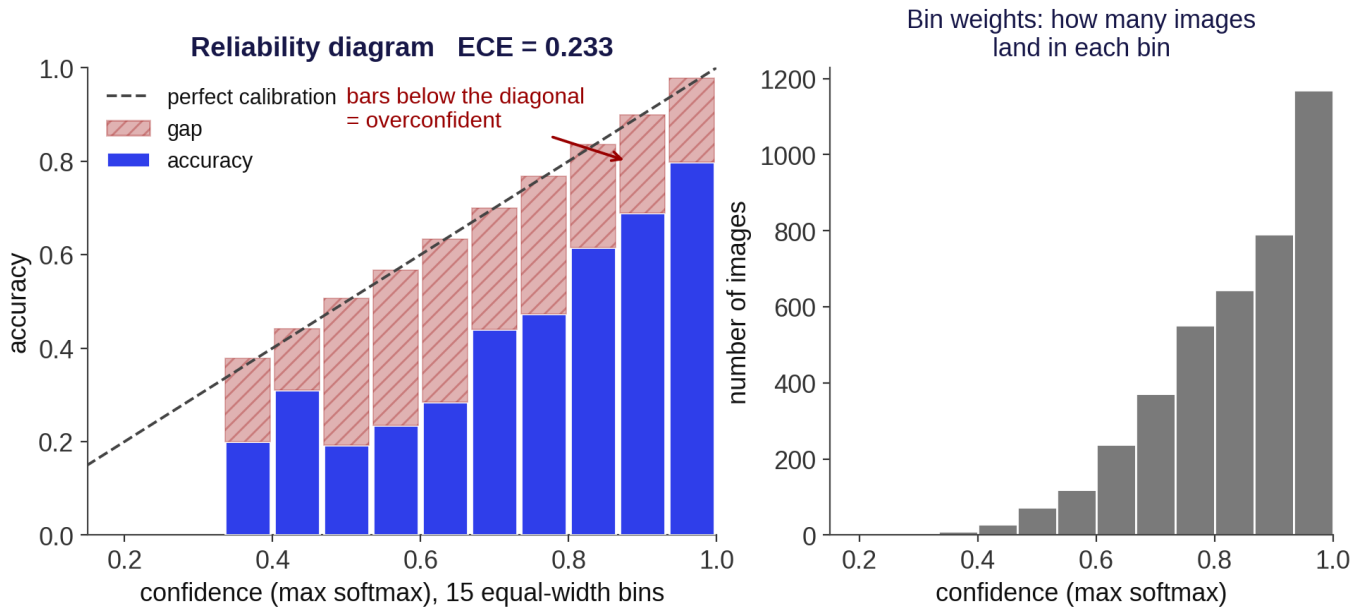
Metric 1 – Expected Calibration Error (ECE)

When you say 80 %, are you right 80 % of the time?

Bin the images by their confidence (max softmax) into 15 equal-width bins. In each bin compare average confidence with actual accuracy.

ECE is the weighted mean of those gaps, weighted by how many images fall in each bin. Lower is better; 0 means perfectly calibrated.

On its own it is gameable: a model that predicts 1/57 for every image is perfectly calibrated and perfectly useless



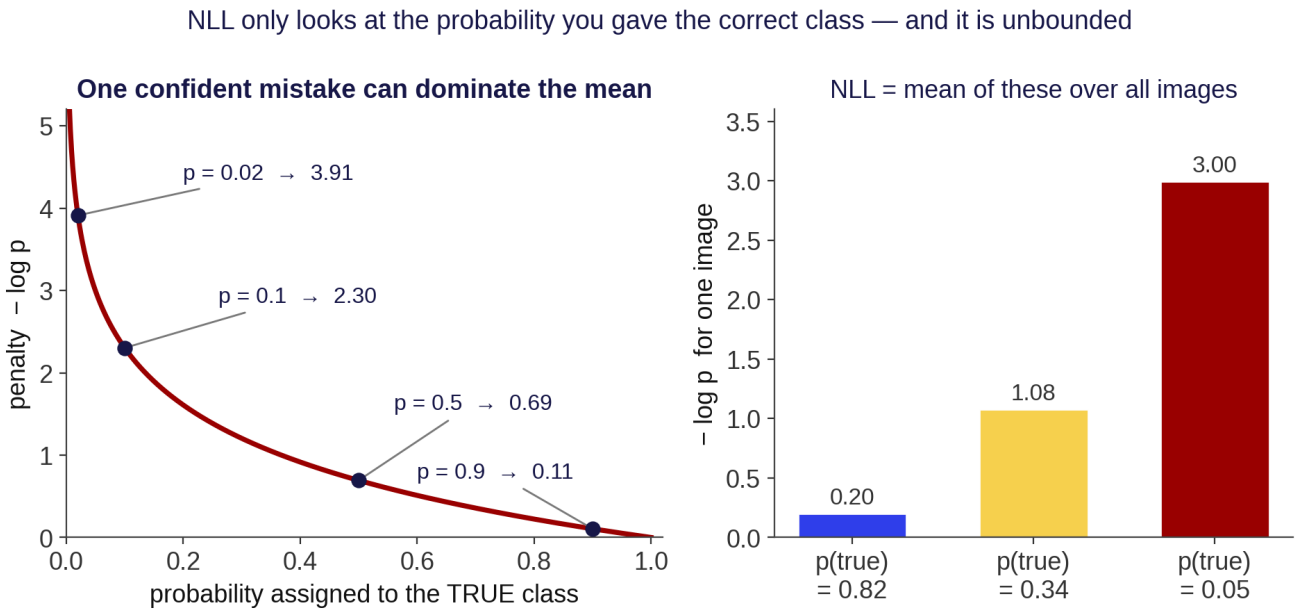
Metric 2 – Negative log-likelihood (NLL)

How much probability mass did you put on the correct class?

Per image the penalty is −log p(true class); NLL is the mean over the test set. Lower is better.

Unbounded: assign 0.001 to the truth and you pay 6.9 on that one image, which can outweigh dozens of good predictions. Never output a hard 0.

A proper scoring rule — minimised only by reporting your true beliefs. This is also the objective temperature scaling optimises on val.



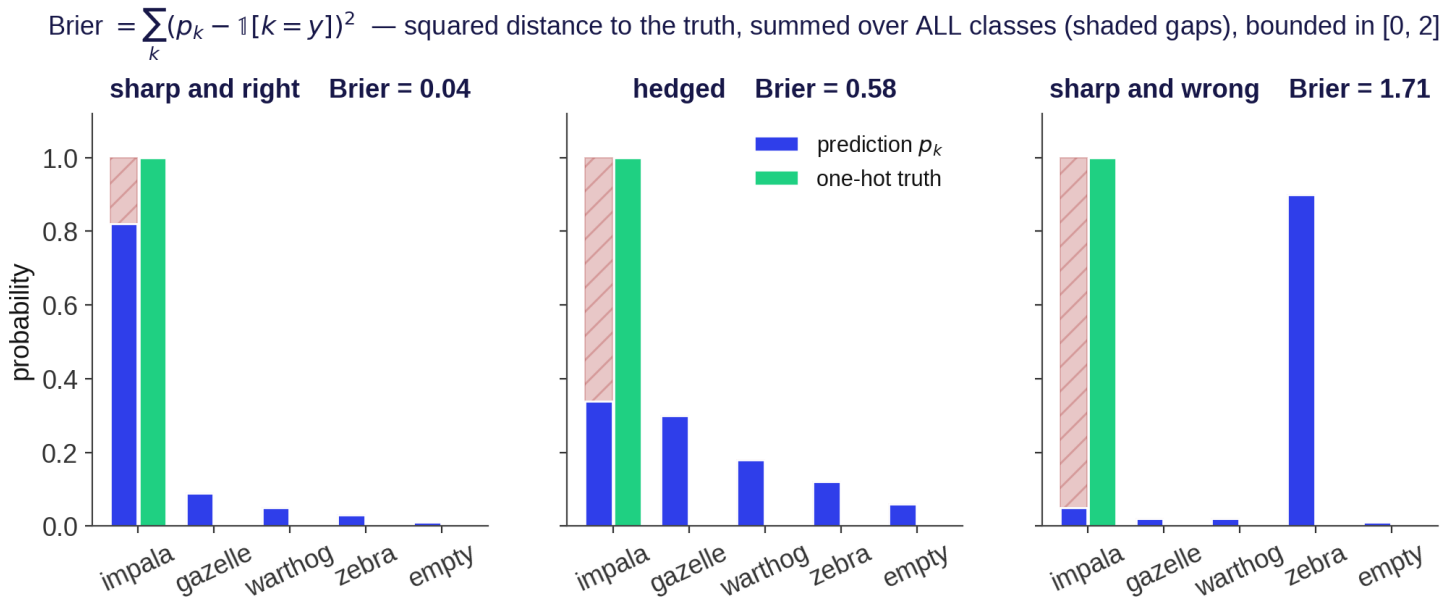
Metric 3 – Brier score

How far is your whole probability vector from the truth?

Squared distance between your K-dimensional prediction and the one-hot label, averaged over images. Lower is better.

Unlike NLL it looks at every class, not only the true one — spreading mass over plausible look-alikes is penalised more gently than piling it on one wrong class.

Also a proper scoring rule, but bounded in [0, 2], so a single catastrophic prediction cannot dominate the average the way it can for NLL.



Metric 4 – Misclassification AUROC

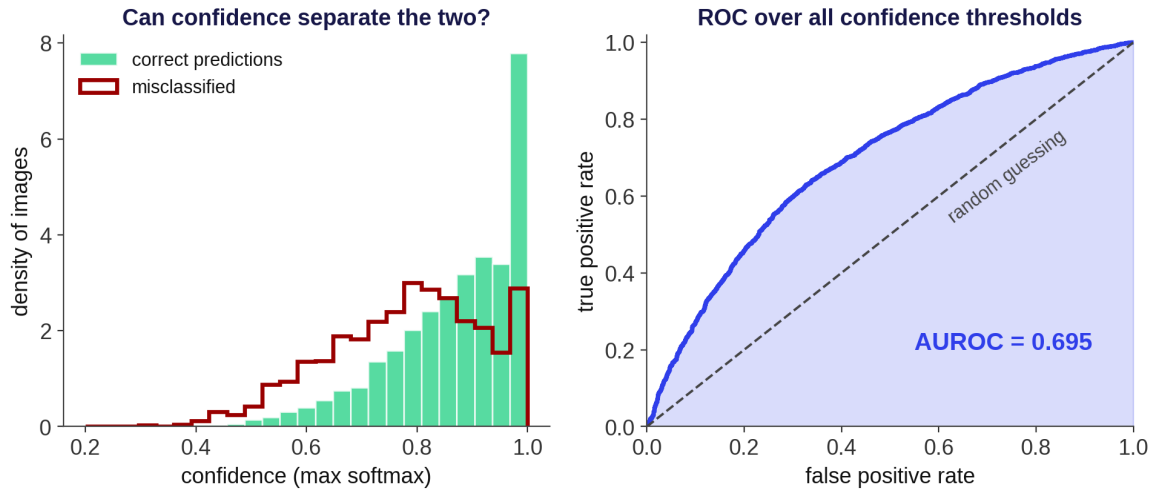
Does the model know which of its own answers to distrust?

Use confidence as a score for "this prediction is correct", sweep every threshold, and measure the area under the ROC curve. Higher is better; 0.5 is random.

Purely about ordering: rescale every confidence with the same strictly increasing function and AUROC is unchanged — it is blind to calibration.

Temperature scaling barely reorders max-softmax, so it moves ECE, NLL and Brier a lot and AUROC almost not at all. To move this one you have to change which predictions look uncertain.

Misclassification AUROC: probability that a randomly chosen correct prediction is more confident than a randomly chosen mistake



Two axes: calibration and discrimination

Where each metric lives — and why no single one of them is enough

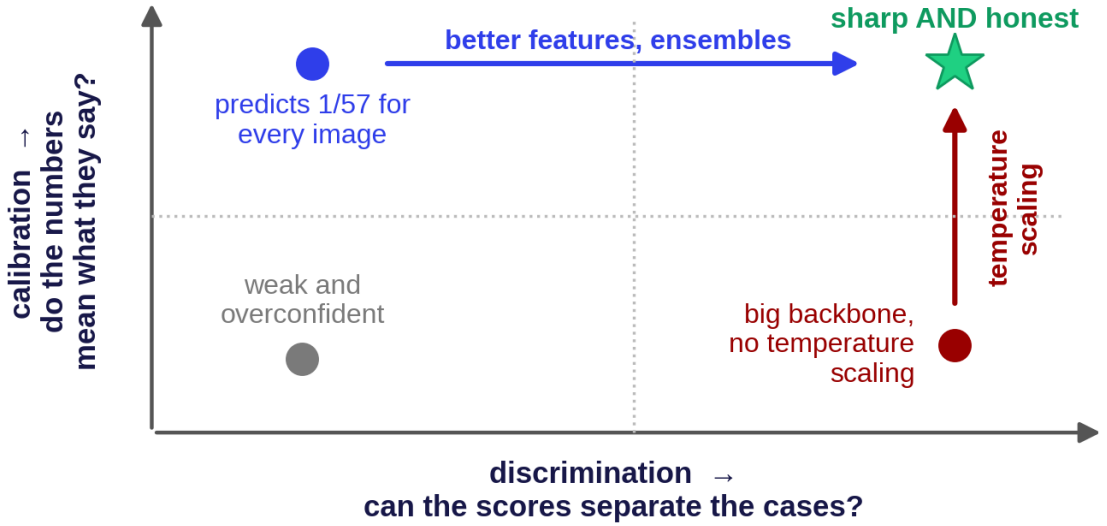
Discrimination: can the scores separate the cases?

Calibration: when you say 0.8, are you right 80 % of the time?

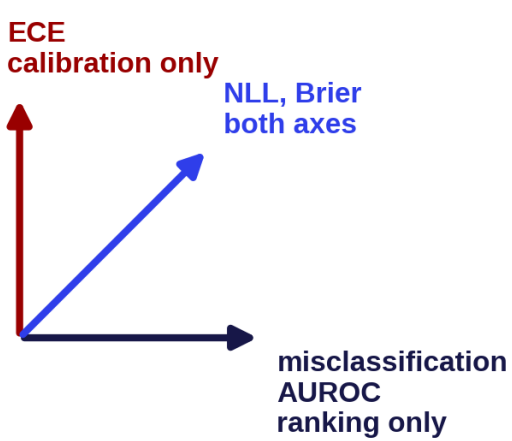
ECE and misclassification AUROC are the orthogonal pair: ECE never sees whether the model can classify at all, AUROC never sees what the numbers mean.

NLL and Brier are diagonal, not orthogonal: every proper scoring rule splits into calibration + refinement — Brier = reliability − resolution + uncertainty (Murphy, 1973), and the log score likewise.

Two axes, not one



What each metric responds to



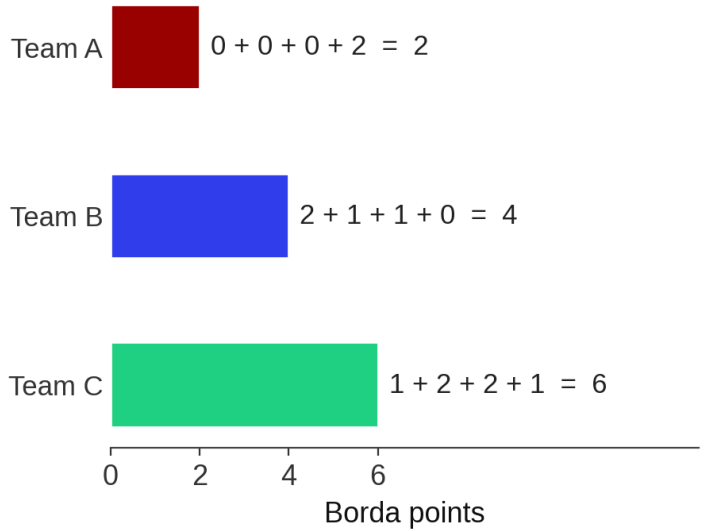
Putting it together – Borda count

You are ranked on each metric separately, the ranks become points, and the points are added up

With N teams, rank r on a metric earns N − r points (here N = 3, so 1st = +2, 2nd = +1, 3rd = +0)

	ECE ↓	NLL ↓	Brier ↓	AUROC ↑
Team A big backbone, no calibration	rank 3 **+0 pts**	rank 3 **+0 pts**	rank 3 **+0 pts**	rank 1 **+2 pts**
Team B smaller model, well calibrated	rank 1 **+2 pts**	rank 2 **+1 pts**	rank 2 **+1 pts**	rank 3 **+0 pts**
Team C ensemble	rank 2 **+1 pts**	rank 1 **+2 pts**	rank 1 **+2 pts**	rank 2 **+1 pts**

Add the points → the leaderboard



Winning one metric outright is worth less than being solidly good at all four. Team A tops AUROC and still finishes last, because its probabilities are badly calibrated.

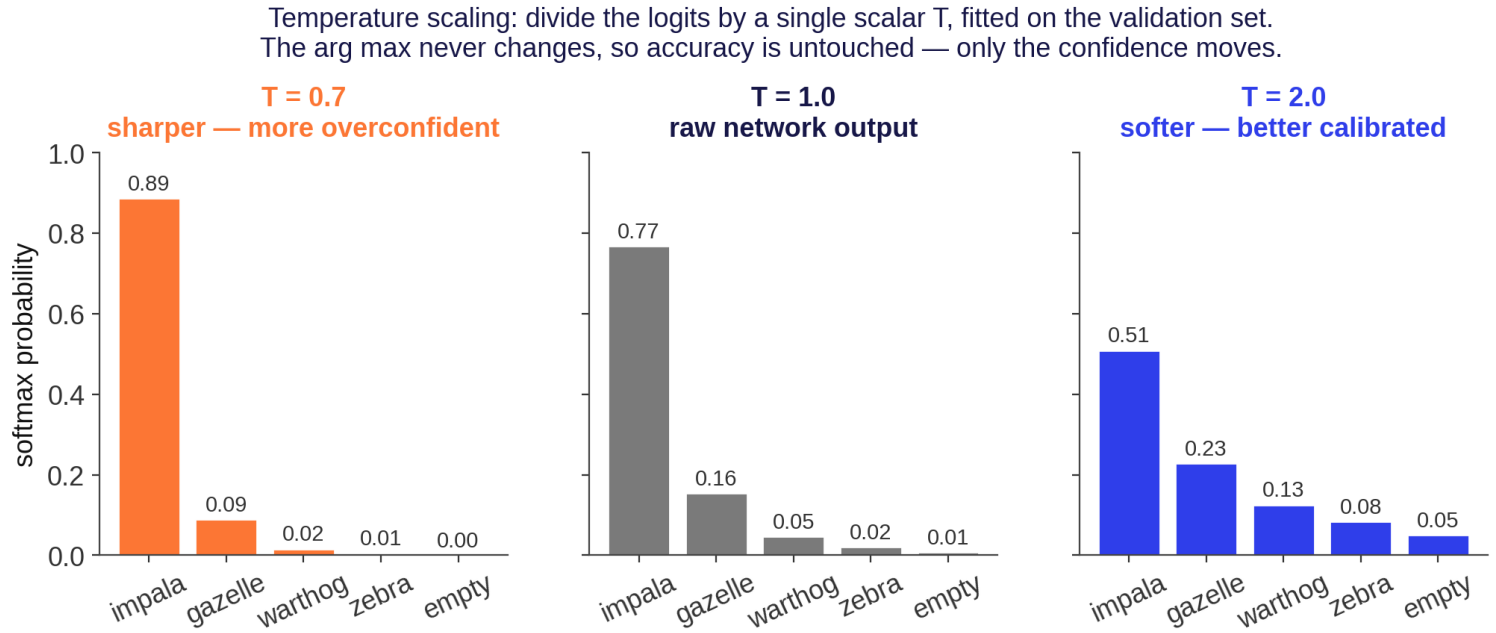
So: no single-metric strategies. A sharp model that is honest about its confidence scores well everywhere.

The baseline you are given

A pretrained CNN backbone with a separate classification head, AdamW (lower learning rate on the backbone, higher on the head), a cosine schedule and early stopping on validation NLL.

After training, a single scalar temperature T is fitted on val with LBFGS on NLL. You get two checkpoints: model.pt (T = 1) and model_temp_scaled.pt.

The backbone is swappable from the command line — the Classifier class exposes backbone, head and embed(x), so you can override forward for MC dropout or ensembling without touching train.py.



Your workflow

- **pip install -r requirements.txt**
  – torch, torchvision, pandas, numpy, pillow, tqdm — the local evaluator is self-contained
- **python -m student.train --data-root … --output-dir runs/baseline**
  – Trains the baseline and fits the temperature on val
- **python -m student.eval --checkpoint … --data-root …**
  – Prints the same four scored metrics the server computes — identical formulas, so iterate here before submitting
- **python -m student.predict --checkpoint … --output submission.csv**
  – Writes one CSV covering test_public and test_private
- **Upload submission.csv to the challenge server**
  – Leaderboard updates from the test_public rows

Do not change the metric formulas in student/metrics.py — the server uses the same ones.

Submission format

```
uid,p_0,p_1,...,p_56
9f2c1ab7d3e04c5f6a8b0d21,0.01,0.72,...,0.004
3ad91f0c8e2b47d6905a1cbe,0.44,0.05,...,0.011
```

One row per uid, across both test_public/images/ and test_private/images/

p_k is the probability of class k in the remapped label space — K = 57, so p_0 … p_56 (see class_mapping.json)

Probabilities must sum to ≈ 1 per row — tolerance 1e-3

Row order does not matter; the server joins on uid

sample_submission.csv in the release is a working template with uniform 1/57 probabilities — it is accepted, just scored very badly

student.predict produces a valid file by default — start from it

Teams

- You should participate in the challenge in teams
- We have pre-made 15 teams
- You should add your name and email to a team on the paper in the conference room
- Each team should have at least one computer with a reasonable GPU, or access to a GPU cluster
- Download the challenge data and place it on that machine before the challenge starts — do not spend the first hour copying images

Challenge dashboard and results

- All test evaluations are done on the challenge website
  – The leaderboard shows all four scored metrics plus the Borda total
- You should submit test scores AT LEAST once per day
- You should also submit the final set scores latest Thursday at 16h!
- Your single submission.csv covers both test splits — the public rows feed the live leaderboard, the private rows decide the final ranking

#	Team	ECE ↓	NLL ↓	Brier ↓	AUROC ↑	Borda
1	Team C	0.031	1.12	0.44	0.86	6
2	Team B	0.024	1.18	0.46	0.81	4
3	Team A	0.161	1.66	0.55	0.88	2

Illustration of the leaderboard layout — the Borda column is what ranks you.

Presentations and results

- The final project presentations and results are on Thursday from 16:30–18:00
- Each team has 4 minutes to present their project with maximum 3 slides
  - How did you design your strategy?
  - Which uncertainty method did you pick, and did it work as expected?
  - What did the four metrics disagree about?
- Finally, the final test results are presented by the organizers

Tips

- Get a submission in early — even the uniform sample submission — so the pipeline is never the thing that blocks you
- Look at id versus ood performance on val separately; the gap is where the challenge is won
- Temperature scaling is already in the baseline. Try fitting it on Brier instead of NLL, or per domain
- Deep ensembles: train N models and average their probabilities — usually the strongest simple win on all four metrics at once
- MC dropout: keep dropout on at eval time and average over several forward passes
- iWildCam is long-tailed — class re-weighting or focal loss can help on the rare species, but check what it does to your calibration
- Never output a probability of exactly 0; clip and renormalise

Rules

- We are not checking for cheating and believe in fair play and that you are here to learn
- We do not recommend you to:
  - Get images or labels from external sites, including the original WILDS iWildCam release
  - Hand-label the test images
  - Change the metric formulas in student/metrics.py
- The goal is to learn about predictive uncertainty — not to squeeze the last half percent out of a classification architecture

How do you get started?

- Go to the summer school website:
  - https://uncertainty.compute.dtu.dk/
- Find the link for the GitHub repo:
  - https://github.com/JosefineVS/UncertaintyChallenge2026
- Clone it, read the README, install the requirements
- Download the challenge data and unzip it into challenge_data/

Scan for the repo

Questions? Ask any of the organizers during the week.

Have fun in the wild!

github.com/JosefineVS/UncertaintyChallenge2026