Distillation

ECBS5200 — Week 6

Six weeks ago you trained a model. Today a 32B teacher trains it.

Where we left off

Week 5 — quantization. Six configurations, five numbers each, one deployment decision.

Tool	What it compresses	What it costs
LLM.int8	Memory at ~equal accuracy	Latency on T4
int4 NF4	Memory + (with right kernel) latency	Some calibration drift on tail
AWQ / GPTQ / FP8	Production-grade compression	Different hardware than you have

The take-home: quantization is a toolbox, not a technique. Match tool to constraint, measure on your hardware.

This week's thesis

> **Distillation transfers specific properties at specific costs. Cheaper alternatives often transfer the same property — but some don't have a substitute.**
>
> The applied ML skill is naming the property you need, picking the cheapest recipe that delivers it, and knowing when no cheaper recipe exists.

Today's shape

Lecture → Lab (80 min) → Homework + memo.

Lab: implement KD loss from scratch, hunt three silent bugs, then compare a vanilla student against a distilled student on the same data. Per-tier F1, per-tier ECE, paired bootstrap CIs.

Homework: pick a deployment scenario, build a recipe shortlist, test two literature claims against your numbers, write a 5-section memo. Prompt 5 of the memo is the capstone synthesis of the term.

Vocabulary you'll hear today

Distillation flavors: soft-target / Hinton-style, hard-label / output-only, synthetic-data SFT, black-box, Stanford Alpaca-style

Loss math: soft target, dark knowledge, temperature `T_d`, KL divergence on softmax

Today's specific recipe: Qwen3-32B teacher (LoRA-fine-tuned, frozen base) → ModernBERT-base student, `T_d = 4`, `α = 0.7`, train+test combined

Mechanism diagnostics: ECE, NLL, JS divergence, per-tier paired bootstrap

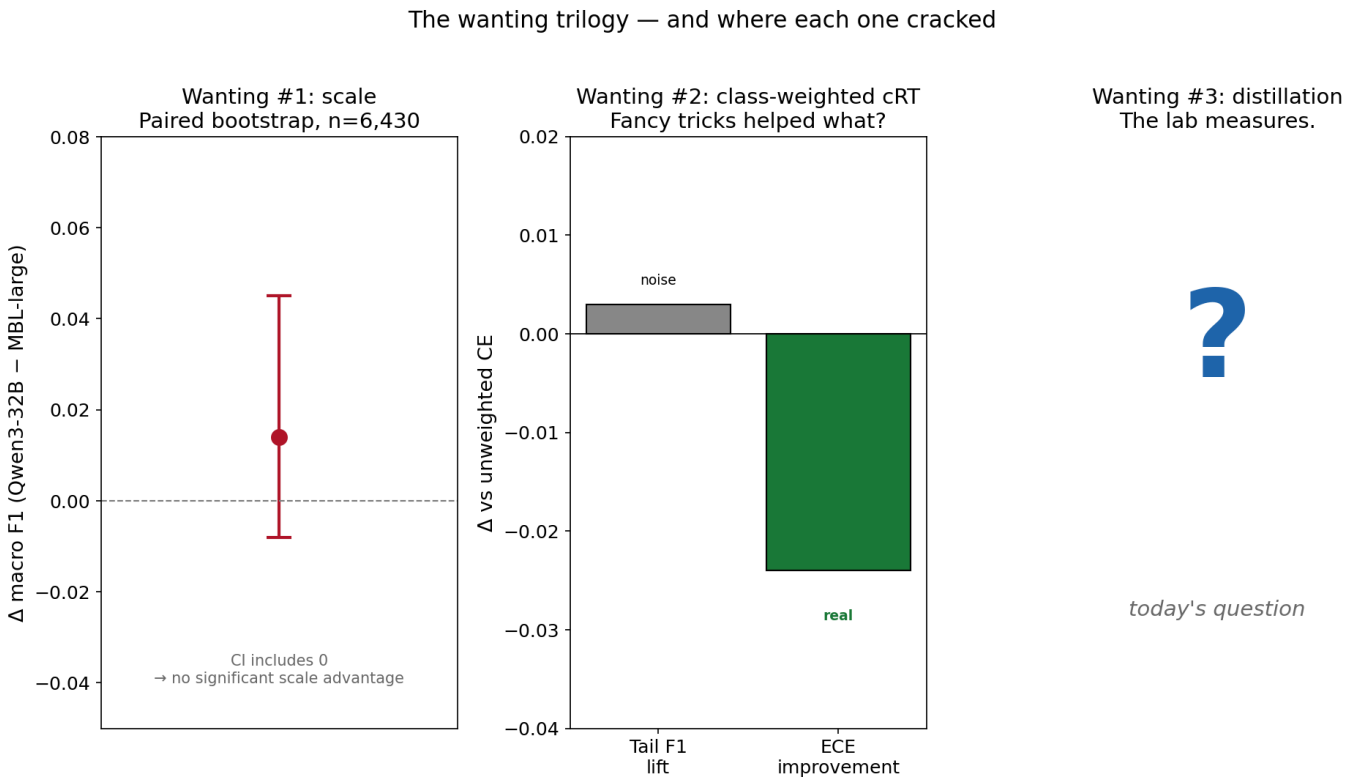
The wanting trilogy

This term you've been chasing one number — macro F1 on a 113-class long-tail task. You've tried three things, and you've been wanting each of them to work.

Wanting #1: scale will fix it. Cracked weeks ago.

Wanting #2: fancy tricks will fix it. Cracked too.

Wanting #3: distillation will fix it. Today's question.



Act 1: What distillation actually is

Before we measure anything, look at the operation.

A student matches a teacher's outputs

A teacher is large and slow. A student is small and fast. The student trains to reproduce the teacher's behavior on the training data.

The simple version — train the student on `(input, teacher_argmax_label)` pairs — works. Sort of. Industry calls this **synthetic-data SFT** or **hard-label distillation**.

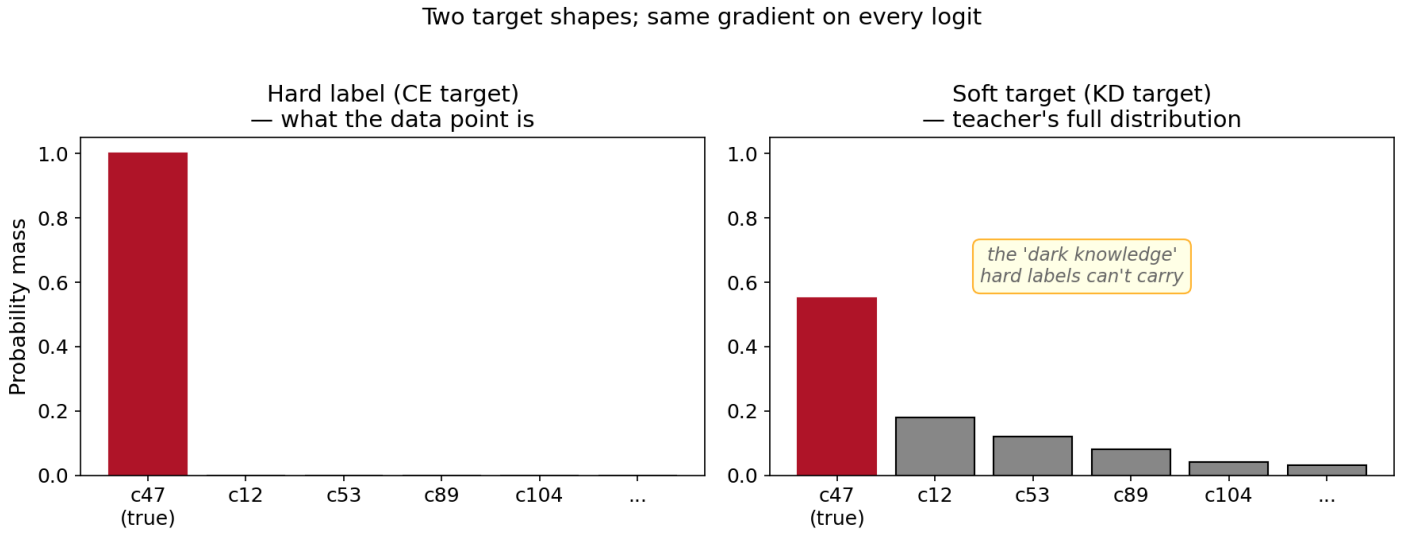
But Hinton et al. 2015 noticed something better. Use the teacher's **full probability distribution** as the target. Not just the argmax.

Two target shapes; same gradient on every logit

Cross-entropy with a hard label points all the gradient toward making the true class confident.

Cross-entropy with the teacher's soft distribution points the student at the teacher's *relative* probabilities across all 113 classes.

Both losses produce gradients on every logit. The difference is the **target shape**.



The KD loss

$$\mathcal{L}_{\text{KD}} = \alpha \cdot T_d^2 \cdot \text{KL}\left(\sigma\left(\frac{t}{T_d}\right) \middle\| \sigma\left(\frac{s}{T_d}\right)\right) + (1-\alpha) \cdot \text{CE}(s, y_{\text{hard}})$$

- `t`, `s` — teacher and student logits
- `T_d` — distillation temperature (softens both distributions before KL)
- `α` — weight on the soft-target term; `(1–α)` on hard labels
- `T_d²` — cancels the `1/T_d²` gradient scaling that softening introduces

Direction: `KL(teacher || student)` — teacher is the target, student is the approximation. The loss penalizes the student for failing to place probability mass where the teacher does — training the student toward the teacher's full probability shape, not just the argmax.

Temperature: what `T_d` actually does

`T_d`	Softmax behavior	Information transferred
1	Sharp; close to argmax	Mostly the top-1 class
4	Moderately soft	Top class + relative shape of next several
8	Very soft, near-uniform	Spread across many classes
∞	Uniform	Nothing

Today's recipe: `T_d = 4`. Homework sweeps `T_d ∈ {1, 4, 8}` against `α ∈ {0.7, 0.9}`.

The teacher we use

Qwen3-32B + LoRA + temperature scaling. Same architectural family as the Qwen 0.5B / 1.5B / 3B decoders you compared against the encoder in Week 3 — just a much bigger member of the family.

Spec	Value
Base model	Qwen3-32B (decoder)
Adaptation	LoRA rank 16 (~3.2M trainable params)
Training data	original train + test split combined (79,278 examples)*
Calibration	post-hoc temperature scaling, `T = 1.25`
Val macro F1	**0.322**
Val ECE	**0.021**

* "test" here means *the original third split we repurposed as additional labeled training data*. **Val** remains the held-out comparison set for this experiment. Don't do this with a true final test set in the wild.

The base model is 32B parameters. Only ~0.01% of them changed during training.

Why students don't load the teacher

A 32B model needs ~64GB of VRAM in bf16. Your T4 has 16GB.

Solution: precompute the teacher's logits once, host as a public dataset.

```
hf_hub_download(repo_id="earino/ecbs5200-week6-teacher-logits",
                repo_type="dataset",
                filename="train_test_logits_qwen3_32b_canonical_final.npz")
# 18.6 MB. fp16. (79,278 × 113) array.
```

The KD loss reads from this array per batch. The student sees the teacher's distribution **without ever loading the teacher**.

What you'll measure today

Same student, same data, same hyperparameters, same seed. Only the loss function differs.

- **Vanilla student:** `CE(student_logits, hard_labels)`
- **Distilled student:** `α · KL(teacher || student) · T² + (1−α) · CE(student_logits, hard_labels)`

The cleanest possible isolation of *"what does distillation specifically transfer."*

The lab pre-computes both for you. You analyze.

Predict, then observe

Today's lab is built around four predictions:

1. **Tail F1.** What macro F1 does a 32B teacher get on the rarest 53 classes?
2. **Distribution shape.** Which tier has the most peaked teacher distribution?
3. **Where does KD's F1 lift land?** Head, mid, tail, or evenly?
4. **Does ECE follow the same per-tier pattern as F1?**

Write your prediction. Run the cell. Reconcile *before* opening the reveal.

The memo rewards specific corrections, not lucky guesses.

Two transfer axes — foreshadowed

By the end of the lab you'll have measured **two different things** the distilled student inherited from the teacher.

One is **capacity** — argmax decision quality on each input (F1, accuracy).*
One is **calibration** — the probability shape over all 113 classes (ECE, NLL).

The lab will tell you whether they decouple, and on what tiers.

** We're using "capacity" loosely to mean "argmax decision quality," not the standard ML usage of "representational power" (param count). When the rubric says "capacity transfer is data-bounded," it means F1 transfer.*

Act 2: Distillation in the news

You've read about this. Now we get specific.

Reading "distillation" stories with precision

Three documented allegations since 2024 — none litigated:
DeepSeek vs OpenAI (Jan 2025) · Anthropic vs DeepSeek/Moonshot/MiniMax (Feb 2026, 16M+ exchanges alleged) · OpenAI memo to Congress (Feb 2026)

Common pattern: API queries → train on completions. Called "distillation" in every headline.

Three questions to ask of any "distillation" allegation:

1. **Logits access, or only API completions?** Closed APIs don't expose full logits → not Hinton-style.
2. **Behavioral evidence or operational?** "Model self-identifies as ChatGPT" vs "logged proxy clusters."
3. **Filed in court, or press / policy posture?** None of these has been adjudicated.

What journalists call "distillation"

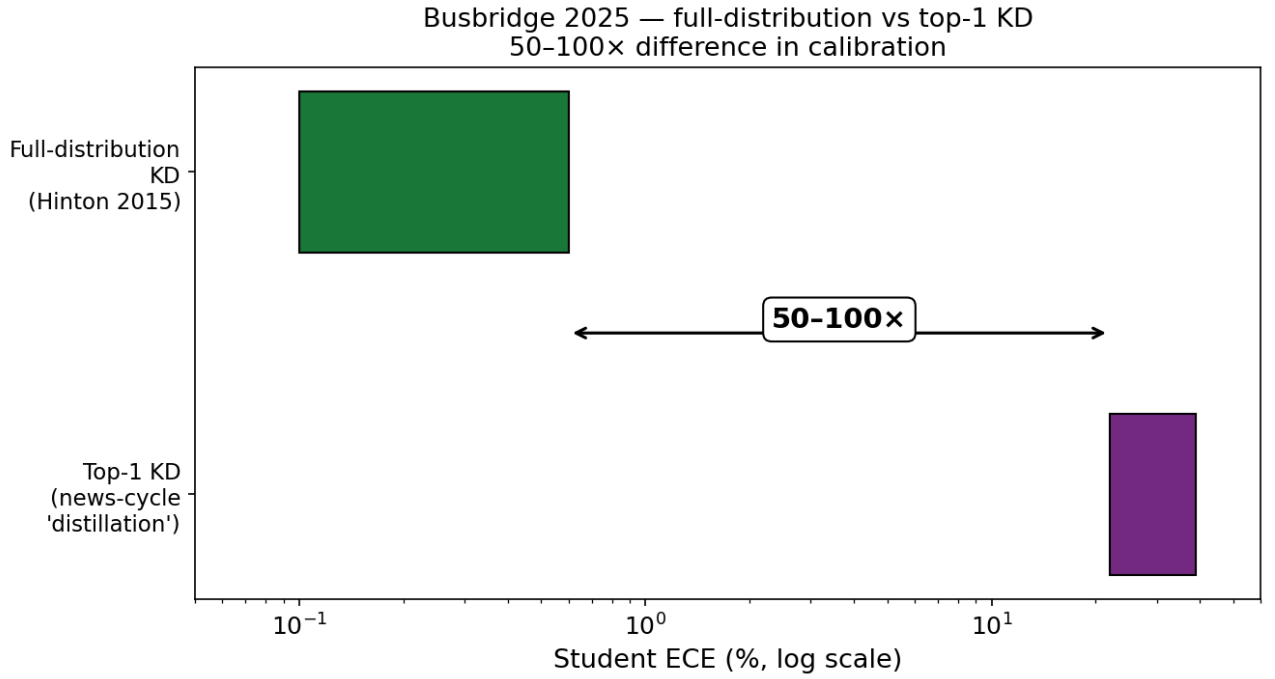
Name	Requires	Used here?
Hinton-style KD — student matches full softmax	Logits access	**No** — closed APIs don't expose full logits
Top-k KD — match top-k logprobs	`logprobs` API param	Possible, slow, no allegations specifically claim
Hard-label distillation — train on teacher's argmax / sampled output	Just API access	**Yes — what every "distillation attack" actually means**
Synthetic-data SFT — teacher labels/rewrites unlabeled data	API access	Yes (often blended)
CoT trace harvesting — query for reasoning traces	API + reasoning model	The Anthropic / Gemini-trace allegations

Busbridge 2025 — the load-bearing measurement

Busbridge et al., ICML 2025, "Distillation Scaling Laws," §E.8. Same student model, same data. Vary only soft-vs-hard target:

- **Full-distribution KD:** student ECE **0.1–0.6%**
- **Top-1 KD:** student ECE **22–39%**

That's a **50–100× difference in calibration** depending on which target shape you used. The full distribution is what carries calibration. Sampled outputs throw it away.



What you measured today vs the news cycle

The news cycle is mostly about hard-label / synthetic-data SFT. Closed APIs force this, regardless of intent.

You did Hinton-style KD today. You had the teacher's full softmax.

Your homework Part 3 Test A measures what this difference buys you on this dataset. JS divergence between teacher and student, per tier. It's the small-scale version of Busbridge's ECE measurement.

The news cycle's unfalsifiable rhetoric becomes a measurement question.

Needle, May 2026 — synthetic-data SFT in the open

Cactus Compute · 26M params · MIT · weights + code public. `github.com/cactus-compute/needle`

Teacher	Gemini 3.1 Flash Lite (public API)
What crosses the wire	Sampled `(query, tools, answer)` triples — no logits
Student loss	Standard cross-entropy on the triples
Post-training	2B tokens, 45 minutes
Architecture	Encoder-decoder, no FFN, gated residual, INT4 QAT

This is the **synthetic-data SFT** row of the taxonomy table earlier in this act — named, dated, and open.

Your lab vs Needle — same word, two recipes

	Your lab today	**Needle (Cactus, May 2026)**
Teacher	Qwen3-32B, weights on Hub	Gemini 3.1 Flash Lite, behind API
Target shape	Full softmax over 113 classes	Sampled tool-call output
Loss	KL(student ‖ teacher) + CE on labels, $T_d=4$, $\alpha=0.7$	Cross-entropy on sampled output
What transfers	Distribution + argmax	Argmax only
Busbridge prediction	**ECE 0.1–0.6%**	**ECE 22–39%**

Same word. Different recipes. Measurable, different consequences.

Act 3: Three compressions, one ceiling

The term in one slide.

Three compressions, the same recurring pattern

Label compression

Week 4 + 5 pipeline reveal

Preserves:

Tractable supervised learning on 113 classes

Doesn't recover:

17 classes too rare to keep (MIN_CLASS_COUNT=5 cut)

Weight compression

Week 5 quantization

Preserves:

Memory + (sometimes) latency at ~equal accuracy

Doesn't recover:

Calibration drift on tail under int4 (Week 5 finding)

Knowledge compression

Week 6 distillation

Preserves:

?

Doesn't recover:

?

Wanting #1: scale (closed earlier this term)

We trained Qwen3-32B with LoRA on this dataset to test whether scale would break the long-tail ceiling.

Paired bootstrap on val (n=6,430), training data held constant:

Comparison	Δ macro F1 (median)	95% CI
Qwen3-32B vs ModernBERT-large	**+0.014**	**[−0.008, +0.045]**

CI includes 0. No statistically significant scale advantage at this dataset's tail length.

Wanting #2: cRT class weighting (closed)

Class-weighted classifier retraining (cRT) — give rare classes higher loss weight.

Outcome	Magnitude
Tail F1 lift over plain CE	**~+0.003** (within noise)
ECE improvement	**~−0.024** (real)
What temperature scaling alone would buy	**~the same ECE improvement**

Class weighting bought calibration, not capacity. And post-hoc temperature scaling matched it without the recipe.

Wanting #3: distillation (today's question)

If scale didn't work and fancy losses didn't work, what about training a small student to inherit a big teacher's behavior?

The hopeful frame:

> Maybe the long-tail data ceiling that bounds *training* a model can be sidestepped by *transferring* a property from one that doesn't have the ceiling.

The skeptical frame:

> If the teacher itself hits the ceiling on tail, what could it possibly transfer?

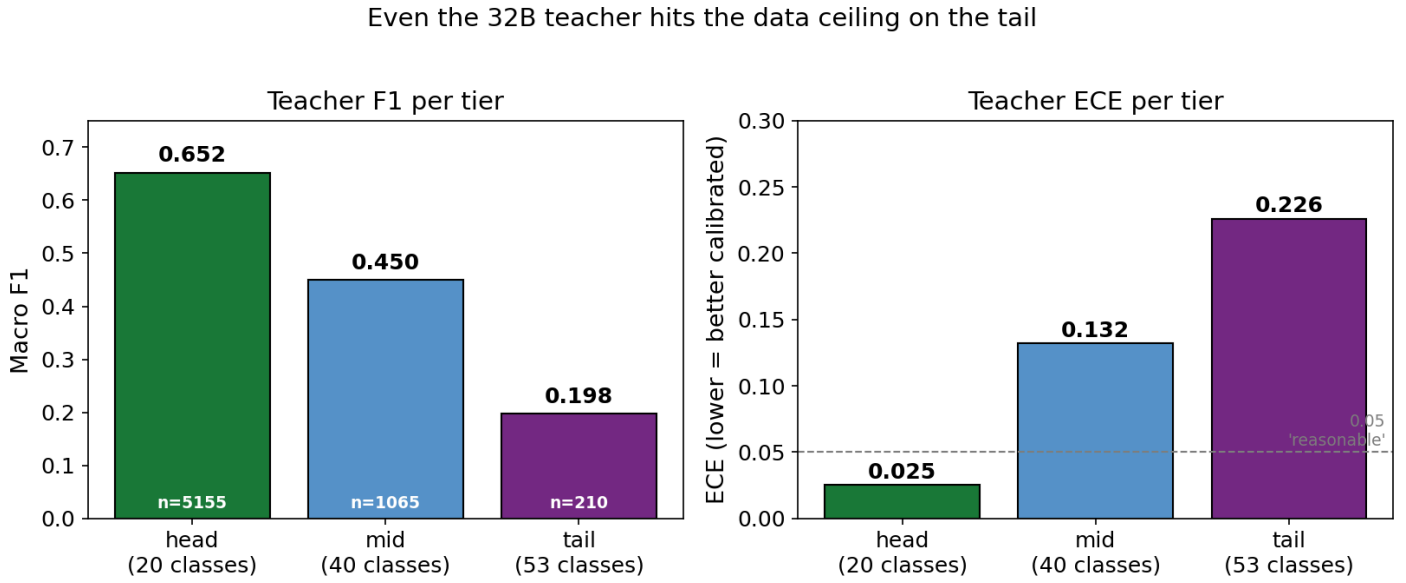
The lab settles it. We measure both arms on the same val set with paired bootstrap CIs.

Even the teacher hits the data ceiling on the tail

Teacher per-tier F1: head 0.652 / mid 0.450 / **tail 0.198**.

A 32B-parameter, carefully-trained, post-hoc-calibrated teacher gets **0.198 macro F1** on the rarest 53 classes — far below its head and mid tiers, and well below what you'd ship.

Why this matters: KD should not be expected to *systematically* transfer tail capacity beyond the teacher's own weak tail performance. Any large tail gain from KD would be surprising and would need careful validation.



What scale does buy you (small but real)

Same paired bootstrap, but compare ModernBERT-base trained on **train only** to the same model trained on **train+test**:

Setup	Macro F1
Week 1 baseline (train only)	0.209
Week 6 vanilla (train+test)	**0.264**

Δ +0.055 from data composition. Not from scale, not from KD — just from showing the model more representative data.

Where each finding leaves the term

Wanting	Verdict	What we measured
#1: scale will fix tail	**Cracked**	Paired bootstrap CI [−0.008, +0.045]
#2: fancy tricks will fix tail	**Cracked**	cRT bought calibration, not capacity
#3: distillation will fix tail	**The lab measures today**	Per-tier F1 + ECE, paired bootstrap

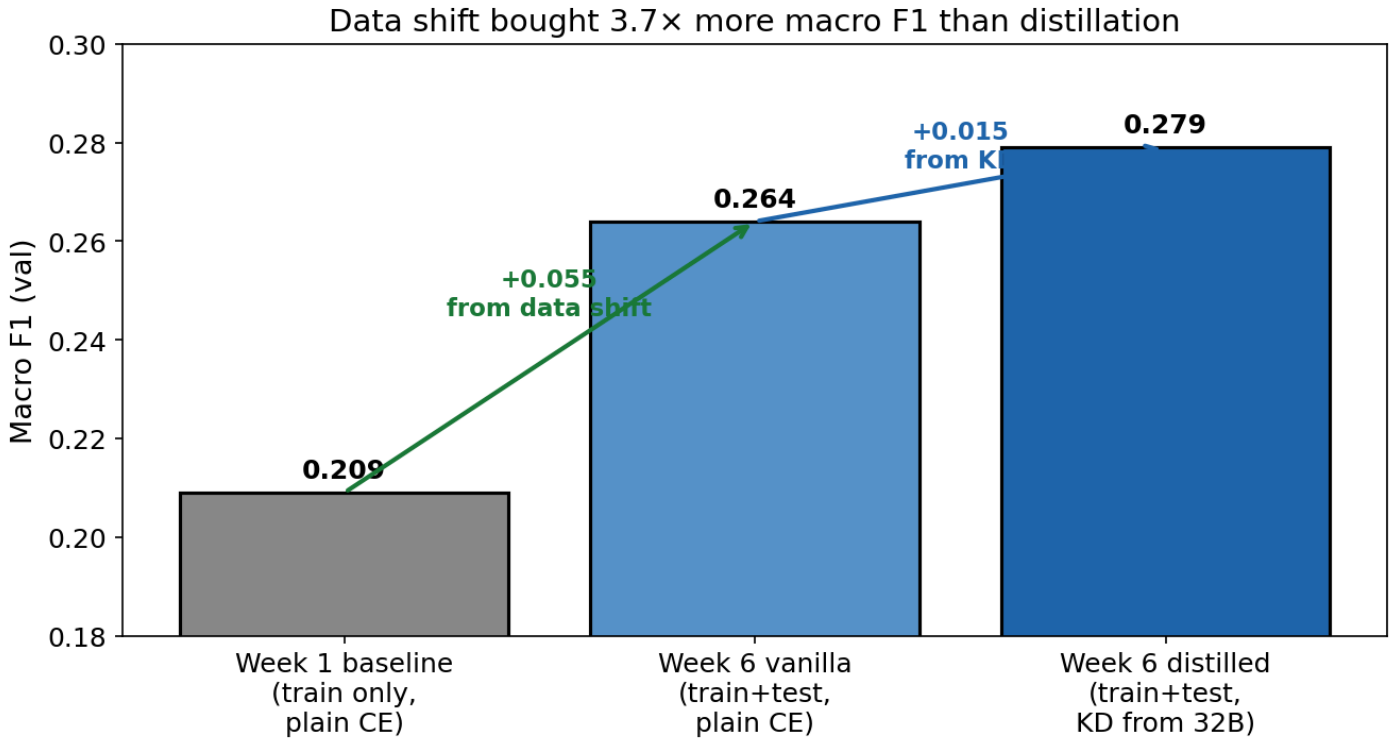
Pattern across the three: every "easy" intervention has come up against the same wall — too few examples per tail class for any technique to escape.

The data confound, made explicit

Lift	Magnitude
Week 1 → Week 6 vanilla (data shift)	**+0.055**
Week 6 vanilla → Week 6 distilled (KD)	**+0.015**

Data shift bought 3.6× more macro F1 than distillation.

The right question for the memo isn't "did distillation help?" — it's "*given a fixed data budget*, what cheapest recipe transfers the property your scenario needs?"



Act 4: What you'll measure

Predict-then-observe, four times. Then defend.

The lab in one slide

80 minutes, three acts:

- **Act 1 (~20 min):** meet the teacher, predict tail F1, look at probability shape
- **Act 2 (~30 min):** implement KD loss, hunt three silent bugs, compare distilled vs vanilla per tier — F1 and ECE
- **Act 3 (~30 min):** threshold/coverage analysis + deployment artifact

Four predict-then-observe cycles. One paired bootstrap (~15 sec on T4).

The bug hunt — what to expect

A colleague sends you their KD loss. Loss decreases during training. The student's calibration looks weird.

Three silent bugs. None crash. All three change what the student learns.

```
def colleague_kd_loss(student_logits, teacher_logits, hard_labels, T_d, alpha):
    """[BUGGY]"""
    kl_term = F.kl_div(
        F.softmax(student_logits / T_d, dim=-1),       # bug?
        F.softmax(teacher_logits / T_d, dim=-1),
        reduction="batchmean",
    )                                                   # bug?
    ce_term = F.cross_entropy(student_logits, hard_labels)
    return alpha * ce_term + (1 - alpha) * kl_term      # bug?
```

Vague hint: two of the three are in the KL term.

What CIs let you say (and what they don't)

Per-tier paired bootstrap on val (n=6,430 total; head 5,155 / mid 1,065 / tail 210).

A CI that excludes 0 → "the effect is real at this sample size."
A CI that includes 0 → "we cannot distinguish this effect from zero."

NOT: "the effect is zero." Just: "we don't have the data to call it."

For your memo: *"the lift is X with CI [Y, Z], inside the noise floor"* is the right form.

Why we precomputed both arms for you

Training one student on T4: ~80 minutes.

Two arms × 80 min = 160 min. Doesn't fit in 80-min lab.

So you load val predictions, not weights. The Hub repos give you fp16 logits + argmax preds + tier assignments per example. Instant.

You're paying for analysis time, not training time.

The §3d artifact: deployment decision

By the end of class: pick one of three scenarios. Fill in every field with a specific value from a table you computed. Defend in writing.

The artifact has three structural rules:

1. Every threshold and coverage value comes from your own §3c table
2. The recipe choice references at least one number from §3a, §3b, or §3c
3. Names one specific constraint that would flip your decision

The form of the answer matters more than the specific recipe.

Act 5: The deployment lens

Calibration is the second axis. Match metric to deployment.

Recap: ECE and NLL are different lenses

Metric	What it measures	When it matters
ECE	Top-1 confidence calibration	Confidence-threshold routing (does 80% confidence mean 80% accuracy?)
NLL	Full 113-dim distribution calibration	Ensembling, downstream Bayesian inference, second-class predictions

One number cannot summarize calibration. Different deployments care about different parts of the distribution.

The skeptic's question

> *If KD's main effect is calibration, and post-hoc temperature scaling fits a single parameter for free, why bother with KD?*

This is a real question. The Han Guo 2021 paper (RepL4NLP) argues KD is "essentially a calibration regularizer" — and that temperature scaling reproduces most of what KD buys.

Your homework Part 3 Test B tests this on this dataset. And the answer turns out to be metric-specific.

Threshold and coverage — the operational story

A common deployment pattern: auto-route only when confidence ≥ T. Below T, escalate to human review.

The policy works only if confidence is meaningful.

Threshold T	What you trade	What you need
Lower	More volume auto-routed	Tolerance for confident-wrong errors
Higher	Higher accuracy on auto-routed	Better-calibrated confidence (or you escalate everything)

A poorly-calibrated model has examples that *look* confident and are wrong. Threshold filtering doesn't help.

Three deployment scenarios in the homework

Scenario	Hard constraint	Primary metric	Likely winning recipe
A — High-throughput batch triage	<10 ms/ex on T4; 10k QPS	Macro F1 + throughput	Vanilla + post-hoc T (fastest, calibrated enough)
B — Regulated escalation review	Calibrated probabilities for human review	ECE + NLL	KD or KD + post-hoc T (full distribution matters)
C — Long-tail rare-class monitoring	Tail-class detection critical	Tail F1 + tail calibration	No recipe likely wins (data ceiling)

The homework forces you to pick **one** and commit. Different scenarios pick different recipes.

The wildcard slot

In Part 2 of the homework, you list 4 measurable recipes — *and one wildcard you don't run*.

The wildcard is a config you propose from first principles and predict where it would land. Examples:

- `T_d = 16` (heavier softening than the grid extreme)
- `α = 0.5` (lower KD weight than 0.7)
- Distilled + post-hoc temperature scaling
- Vanilla + label smoothing

You justify the prediction *mechanistically*. You don't run it.

Cross-week reach

Prompt 5 of your homework memo is the capstone synthesis — it integrates all six weeks into one defended deployment recommendation.

Possible cross-week recipes the homework Prompt 4 asks about:

- **Quantized distilled student** — does Week 5's int4 calibration drift survive KD?
- **Vanilla + temperature scaling + int8** — cheap calibration + cheap memory
- **Distilled with merged labels reverted** — does adding back the dropped 17 classes change anything?

You only measured one of these. Prompt 5 defends the recipe you'd actually ship.

Closing

What you do today, and what the term gave you.

Lab and homework — logistics

Lab in class today (~80 min).

- `notebooks/week6/week6_lab.ipynb`
- Predict-then-observe rhythm, four cycles
- Implement KD loss + bug hunt + per-tier comparison
- §3d deployment artifact due before you leave

Homework after class (~5 hours).

- `notebooks/week6/week6_homework.ipynb`
- 4 parts: diagnose, shortlist, literature tests, memo
- Memo + HTML upload to Moodle — **see Moodle for the exact deadline.** The Week 6 memo is due before the final exam.

Today in one sentence

> **Distillation transfers what's in the teacher's full distribution. The cheaper alternative is post-hoc temperature scaling. Match metric to deployment, and know what each recipe leaves behind.**

You will leave today with a measurement-grounded answer to "what does KD actually transfer," vocabulary to read the news cycle on this with precision, and a defended engineering position from this term's work.

What the term gave you

You can now do something most working ML engineers cannot:

- Diagnose a model **per tier**, not just aggregate
- Defend a number with a **bootstrap CI**, not a point estimate
- Distinguish **what your dataset can support** from what your method should do
- Read a paper or a press release and ask **"what did they actually measure?"**

You'll forget specific (`T_d` , `α`) values. You won't forget the discipline.

Thank you

Lab starts in 5 minutes.

Questions? Anything from the term that didn't land?

If you have time after lab — readings/week6/ has 12 papers. Hinton 2015 is short and beautiful. Stanton 2021 is the optimization-difficulty story. Busbridge 2025 is the LLM-scale measurement of everything we did at small scale.