Towards Resource-Efficient LLMs: End-to-End Energy Accounting of Distillation Pipelines

Katherine Lambert, Sasha Luccioni

University of Toronto, Huggingface, Sustainable AI Group

MOTIVATION

- We live in a world of fast LLM deployment, rising GPU demand, datacenter expansion, increasing energy costs
- Distillation is widely claimed to be a **cheaper, greener** alternative to direct training
- But energy claims almost never account for **teacher-side costs**: logit caching, synthetic data generation, evaluation
- Prior work reports **student-only savings -** the upstream pipeline is ignored

Result: cost-efficiency claims are unsubstantiated, and the true cost is hidden

CONTRIBUTION

Contrary to common belief, found that **distillation is NOT always cheaper** - full pipeline energy tells the whole picture

📐 **Energy accounting framework** 📐
accurate end-to-end, stage-wise measurement of distillation pipelines via NVML GPU telemetry

📊 **Controlled benchmark** 📊
1B / 7B / 13B student sizes distilled from a 32B teacher; energy–quality tradeoff Pareto frontiers

🔧 **Open-source harness + design rules** 🔧
code for measurement, reusable protocol and break-even conditions to help determine when distillation is truly energy-efficient

METHODS

Baseline → no teacher-side cost
KD → Logit Caching
Synthetic SFT → Response Generation

Stage Decomposition:

Code + Harness

$$E_{\text{total}} = E_{\text{prerun}} + E_{\text{teacher}} + E_{\text{student}} + E_{\text{eval}}$$

GPU energy via NVML telemetry (sampled every 0.5s): $E_{\text{GPU}} \approx \int_{t_s}^{t_e} P_{\text{GPU}}(t)\, dt.$

Normalized by tokens: J/token = $E_{\text{total}}/N_{\text{tokens}}$

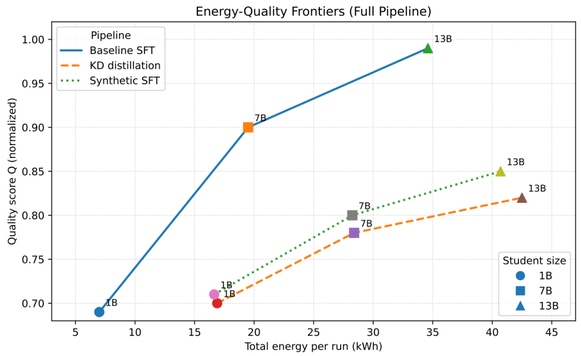
CPU energy obtained via *CodeCarbon estimates*.

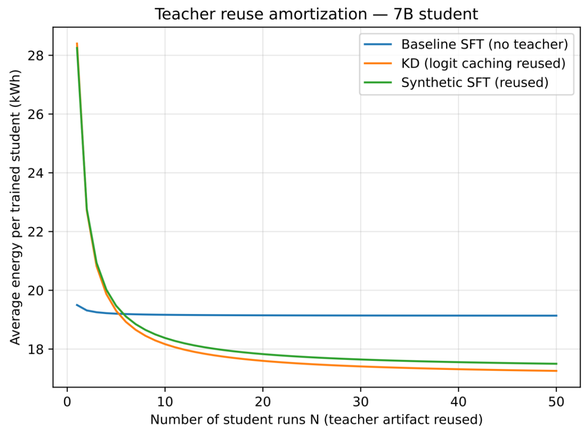
$$E_{\text{total}}^{(\text{stage})} = E_{\text{GPU}} + E_{\text{CPU}}$$

Three pipelines, three student scales *(1B / 7B / 13B)*, single **H100 80GB**.

All runs use fixed hardware, software, tokenizer, and training config for comparability.

RESULTS





Energy–Quality Frontiers *[Figure 1]*

Once teacher costs are included → *KD and Synthetic SFT use far more energy for minimal quality gain*

At 1B: distillation uses *~2.4× more energy* than baseline SFT for marginal Q improvement

At 7B and 13B: *baseline SFT strictly dominates* on the energy–quality frontier

Teacher overhead acts as a *near-fixed cost* - most damaging at small scale

Stage-Wise Breakdown [*Figure 2]*

Teacher Reuse and Break-Even Thresholds *[Figure 3]*

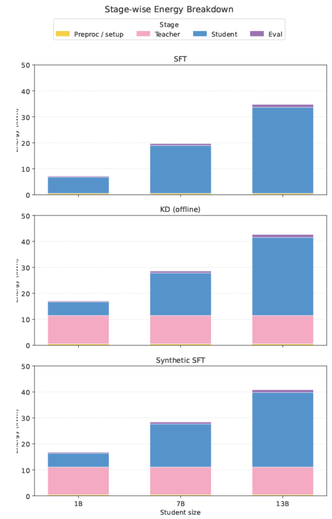
$$N^* = \frac{E_{\text{teacher}}}{E_{\text{student}}^{\text{baseline}} - E_{\text{student}}^{\text{distill}}}$$

Teacher artifact costs amortize as 1/N across reused runs. The break-even reuse threshold:

Pipeline	1B	7B	13B
KD	≈ 10	≈ 5–6	≈ 4
Synthetic SFT	≈ 11	≈ 6	≈ 2–3

Smaller students require **more reuse** to amortize the fixed teacher cost; one-off distillation at small scale is least energy-competitive.

Stage	1B	7B	13B
Prerun	0.12	0.12	0.12
Baseline SFT (no teacher)			
Data preprocess	0.37	0.37	0.37
Student training (SFT)	6.30	18.45	33.15
Evaluation	0.33	0.68	1.08
KD distillation (offline)			
Data preprocess	0.37	0.37	0.37
Logit caching	11.00	11.00	11.00
Student training (KD)	5.20	16.35	30.05
Evaluation	0.33	0.68	1.08
Synthetic SFT			
Data preprocess	0.37	0.37	0.37
Synthetic data generation	10.60	10.60	10.60
Student training (SFT on synthetic)	5.35	16.60	28.65
Evaluation	0.33	0.68	1.08



Takeaways

(1) Distillation is not inherently cheap.
Teacher-side costs can rival student training
(2) End-to-end accounting changes the ranking.
(3) Reuse is the decisive lever. Amortizing teacher artifacts across students, seeds, or sweeps can flip the frontier;
(4) Stage-wise attribution tells you what to optimize.
Don't tune the wrong knob - target the dominant stage for your pipeline and scale.

Design Recommendations

(i) Report full-pipeline energy,
not student-only, with explicit reuse assumptions.

(ii) **Treat teacher artifacts as shared infrastructure**
cached, versioned, and published with metadata, config, and measured production energy. Default to *reuse-before-regenerate*.

(iii) **Use baseline SFT when reuse is low.**
Favor KD or Synthetic SFT when reuse and quality targets are high.