STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 22: Adapting Big Models, and Some Opinions

November 24, 2026

Logistics

- **Today's material (PEFT, RAG, and the opinions) is not on the final exam**
- HW5 due Fri 12/4
- No class Thu 11/26 (Thanksgiving)
- Project presentations: Tue 12/1 and Thu 12/3
  - 4-minute talks; upload slides to Canvas 24 hours before your slot
- Tue 12/8: review lecture; project final report due 11:59 PM on Canvas
- Final exam: Saturday, December 12, 5:05–7:05 PM

Ben Lengerich © University of Wisconsin-Madison 2026

Last Time: Reasoning Models

- Writing out steps = more computation per question (**test-time compute**)
- **RLVR:** replace the learned reward model with a program that checks the answer
- **GRPO:** policy gradient with the group's average reward as the baseline
- **DeepSeek-R1:** longer reasoning emerged from outcome rewards alone; SFT → RL → SFT → RL
- Small models: **distilling** a reasoning model beat running RL on them

Today: Adapting Big Models, and Some Opinions

- **Part 1, technical (~35 min):** adapting a big model cheaply: adapters, LoRA, soft prompts, retrieval
- **Not on the final.** Learn it because you will use it (HW5 Part C implements LoRA)
- **Part 2, opinions (~35 min):** semi-technical questions without settled answers
  - In practice: prompt, retrieve, or fine-tune? Which skills last?
  - Open vs. closed: who gets to build and adapt AI?
  - What society and institutions do you want to build?

Ben Lengerich © University of Wisconsin-Madison 2026

Part 1: Adapting a Big Model Cheaply

Why not just fine-tune everything?

Training takes a lot of memory.

7B-parameter model	Memory to train
Full fine-tuning (Adam, mixed precision)	≈ 16 bytes/param ≈ 112 GB
LoRA (base frozen in bf16)	≈ 2 bytes/param ≈ 14 GB
QLoRA (base frozen in 4 bits)	≈ 0.5 bytes/param ≈ 3.5 GB

plus activations, and a small amount for the trainable part

- **Memory:** gradients and Adam state for every weight
- **Storage:** a full copy of the model for every task or user
- **Forgetting:** updating every weight can erase what the base model knew
- Fine-tuning data is small; the change it needs is probably small too

Where can the new parameters go?

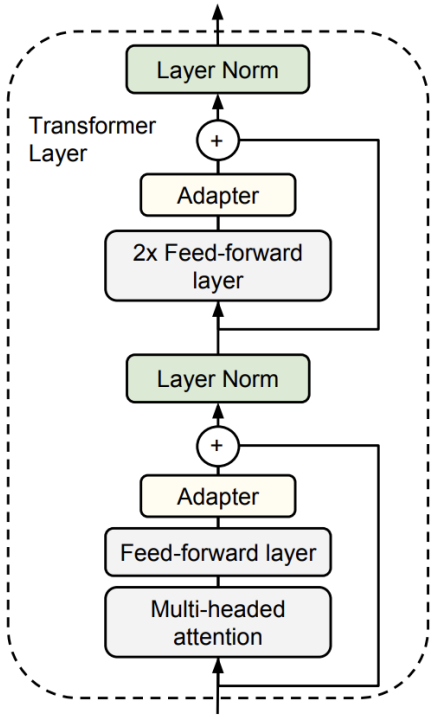
Inside the weights	New modules	In the input	A subset of weights
$W_0 + \Delta W$, with ΔW low-rank (LoRA)	small networks inserted in each block (adapters)	learned virtual tokens (prompt / prefix tuning)	e.g. only the biases (BitFit)

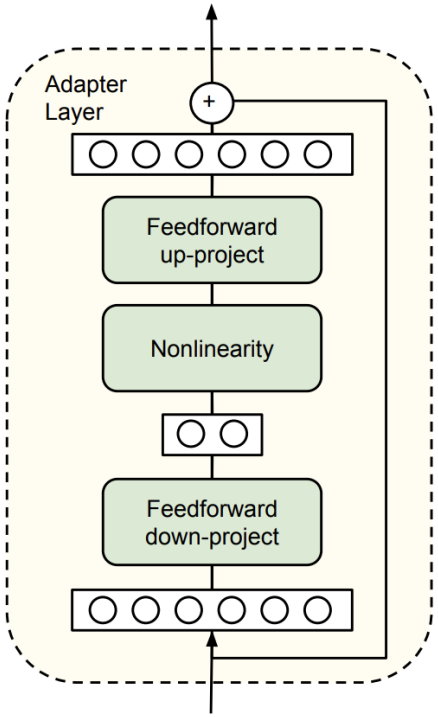
- All freeze the pretrained weights W_0 and train $\lesssim$ **1%** as many parameters
- One frozen base model, many small task-specific add-ons
- The extreme: **in-context learning** trains nothing (Lecture 19)

Adapters: small networks inside each block

$$h \leftarrow h + W_{\text{up}}\, f\left(W_{\text{down}}\, h\right), \qquad W_{\text{down}} \in \mathbb{R}^{m \times d},\ m \ll d$$

- Bottleneck MLP with its **own residual connection**
- Starts near the identity, so training starts at the pretrained model
- GLUE: within **0.4%** of full fine-tuning, adding **3.6%** parameters per task
- Cost: extra layers at inference → **added latency**





Houlsby et al., ICML 2019

LoRA: learn a low-rank change to each weight

$$h = W_0 x + \frac{\alpha}{r} BAx, \qquad B \in \mathbb{R}^{d_{\text{out}} \times r},\ A \in \mathbb{R}^{r \times d_{\text{in}}},\ r \ll d$$

- W_0 frozen; train only A and B
- Hypothesis: the change needed for a new task has **low intrinsic rank**
- Init: A random, **B = 0** → BA = 0, so training starts exactly at the pretrained model
- α/r keeps the update's size steady as you change r

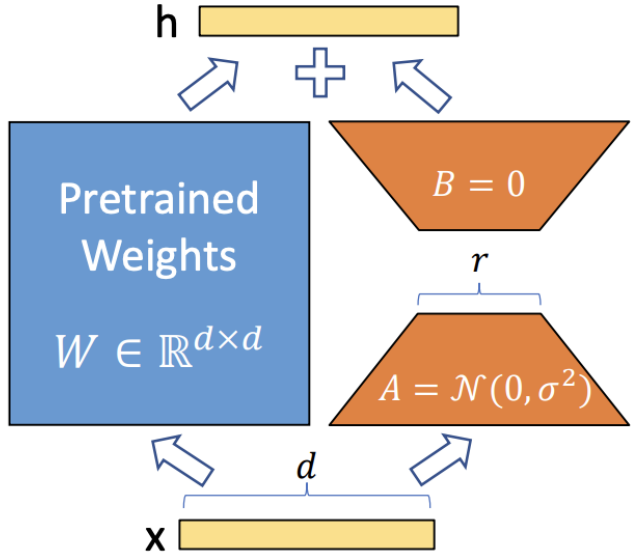


Figure 1: Our reparametrization. We only train A and B.

Hu et al., 2021

Counting LoRA parameters

$$\underbrace{r\,(d_{\text{in}} + d_{\text{out}})}_{\text{LoRA}} \quad \text{vs.} \quad \underbrace{d_{\text{in}}\,d_{\text{out}}}_{\text{full}}$$

One 4096 × 4096 matrix	Trainable	Fraction
Full fine-tuning	16,777,216	1
LoRA, r = 1	8,192	1/2048
LoRA, r = 4	32,768	1/512
LoRA, r = 16	131,072	1/128
LoRA, r = 64	524,288	1/32

- Linear in r and d, not quadratic in d
- GPT-3 175B: **10,000×** fewer trainable parameters, **3×** less GPU memory than full fine-tuning

Merge for free inference

$$W_0 x + \frac{\alpha}{r} B(Ax) \;=\; \left(W_0 + \frac{\alpha}{r} BA\right) x \;=\; W_{\text{merged}}\, x$$

- W_{merged} has the same shape as W_0 → **no extra inference cost** (adapters add layers)
- Same outputs as the unmerged model, exactly
- Swap tasks: subtract one BA, add another
- Or keep them unmerged: one shared base model, many small per-user (A, B) pairs

Ben Lengerich © University of Wisconsin-Madison 2026

Why does low rank work, and what does it cost?

- **Why it works:** small ranks (even r = 1) were competitive when adapting GPT-3
- The base model already has the features; the task needs a **small change in direction**
- Rank r is a **capacity knob**, like a bottleneck

- **What it costs:** on harder targets (code, math), LoRA **learns less** than full fine-tuning
- … and **forgets less** of what the base model could do
- Full fine-tuning's weight changes have 10–100× higher rank than typical LoRA
- HW5: compare Austen loss (learned) with Shakespeare loss (forgotten)

Hu et al., 2021; Biderman et al., TMLR 2024

From hard prompts to soft prompts

Hard prompt (Lecture 19)

- Real tokens: “Classify the sentiment: …”
- Chosen by hand or by search
- Nothing trained

Soft prompt (prompt tuning)

- k learned vectors prepended to the input embeddings
- Need not be any real word
- Trained by gradient descent; model frozen

- Parameters: k × d, e.g. 20 × 4096 = 81,920 for the whole model
- Competitive with full fine-tuning once models reach **billions of parameters**

Lester et al., EMNLP 2021

Comparing the options

Method	What is trained	Extra inference cost	Merge into W_o?
Full fine-tuning	Every weight	None	n/a (a new model)
BitFit	Bias terms only	None	Yes
Adapters	Small MLPs in each block	Extra layers (latency)	No
LoRA	Low-rank B, A per weight matrix	None once merged	Yes
Prompt tuning	k input embeddings	k extra tokens	No
Prefix tuning	Keys/values at every layer	Longer attention	No
In-context learning	Nothing	A long prompt	n/a

Fine-tune or retrieve?

Retrieve (RAG)

- Facts that change or must be cited
- Private documents, recent events
- Update by editing the document store
- No training; weights unchanged

Fine-tune (e.g. LoRA)

- Behavior, style, format
- Domain skills and vocabulary
- Update by training again
- Cheap to serve once merged

Often both: a fine-tuned model that is good at using retrieved documents

Ben Lengerich © University of Wisconsin-Madison 2026

Part 1 summary

- Freeze the big model; train a **small** add-on: $\lesssim$ 1% of the parameters
- **Adapters:** new bottleneck modules; extra inference latency
- **LoRA:** $h = W_o x + (\alpha/r)BAx$; $B = 0$ at init; merges into W_o for free
- **Prompt / prefix tuning:** learn vectors in the input or at every layer
- Trade-off: learns less, forgets less; evaluate both
- **RAG** changes the context, not the weights

Part 2: Opinions

Open vs closed AI…

Should the weights of the next frontier model be released openly?

- Who gains, and who bears the risk?
- Does your answer change if the model is 10× more capable?
- Is training on a closed model's outputs (distillation) fair game?
- **Moral question:** who should get to make this decision for everyone else?

Ben Lengerich © University of Wisconsin-Madison 2026

Moral lenses for today's questions

Consequences	Consent and rights
Who benefits? Who bears the cost?	What did the people affected agree to?

Responsibility	Character
If it goes wrong, who answers for it?	What would you be proud to have built?

Ben Lengerich © University of Wisconsin-Madison 2026

AI makes statistical thinking more important

What a model can find

- Patients on a treatment have worse outcomes
- A shortcut that scores well on the test set
- A confident answer from the data it was given

What it cannot tell you alone

- Whether the treatment causes harm (sicker patients get treated)
- Whether the shortcut survives a new site or an intervention
- Whether the data were a biased sample

The better models get at finding patterns, the more it matters to know which patterns mean something that you want to encode.

Ben Lengerich © University of Wisconsin-Madison 2026

In practice: climb one rung at a time

1. Prompt	2. Retrieve	3. LoRA	4. Full fine-tune
minutes; nothing trained (Lecture 19)	hours; facts that change (RAG)	a GPU-day; behavior, format ($\lesssim$ 1% of weights)	expensive; new skills (and more forgetting)

- **My take:** most teams reach for fine-tuning a rung too early, and knowledge you need to check and keep belongs outside the weights
- Build the evaluation set before choosing a rung; it outlives every model you try
- Move up only when the evaluation shows the lower rung has plateaued

Ben Lengerich © University of Wisconsin-Madison 2026

Which skills get cheaper, which get more valuable?

Getting cheaper

- Writing a model implementation
- Generating experiments
- Producing an analysis
- Writing tests

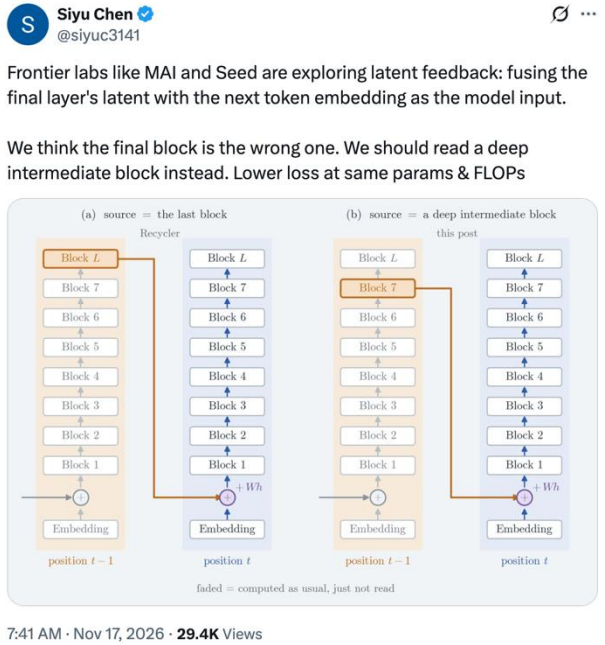
Getting more valuable

- Knowing whether it is the right model
- Designing informative experiments
- Recognizing a misleading analysis (“diagnose the curves”)
- Knowing what correctness means

- **My take:** coding agents write more of the code; you are still responsible for whether the result is right
- Use AI coding tools aggressively, but not to skip building your own judgment

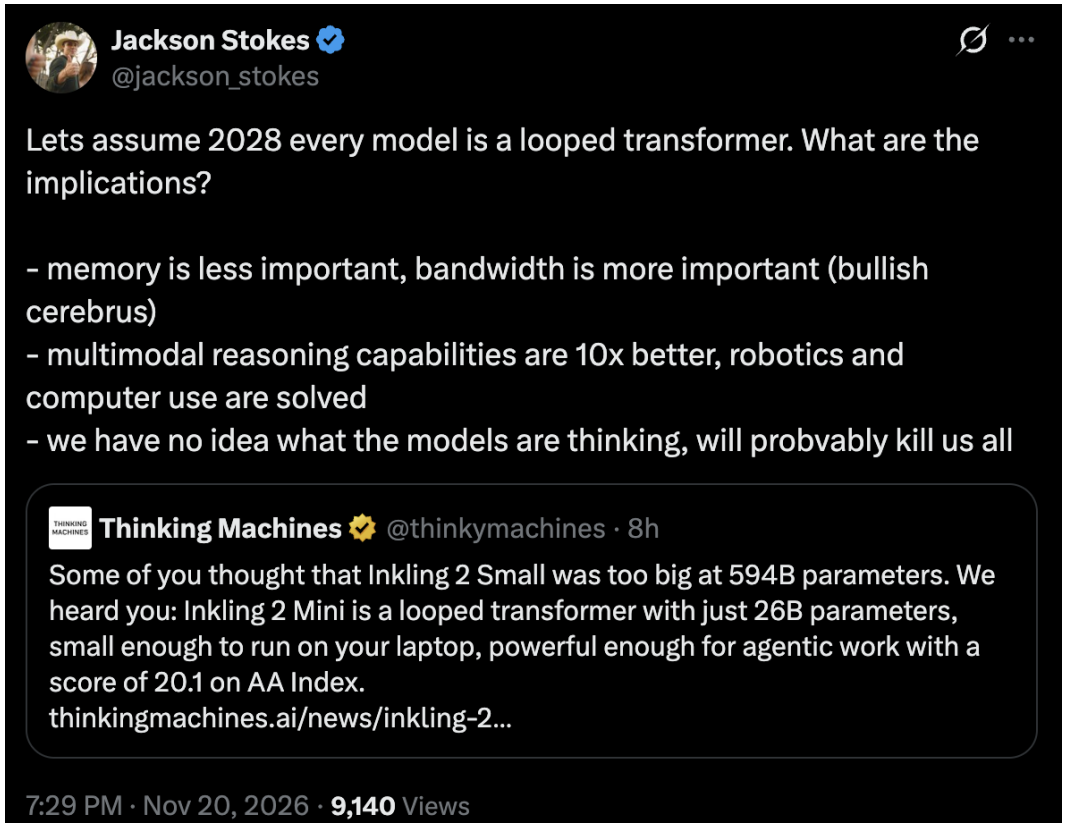
Old ideas keep coming back





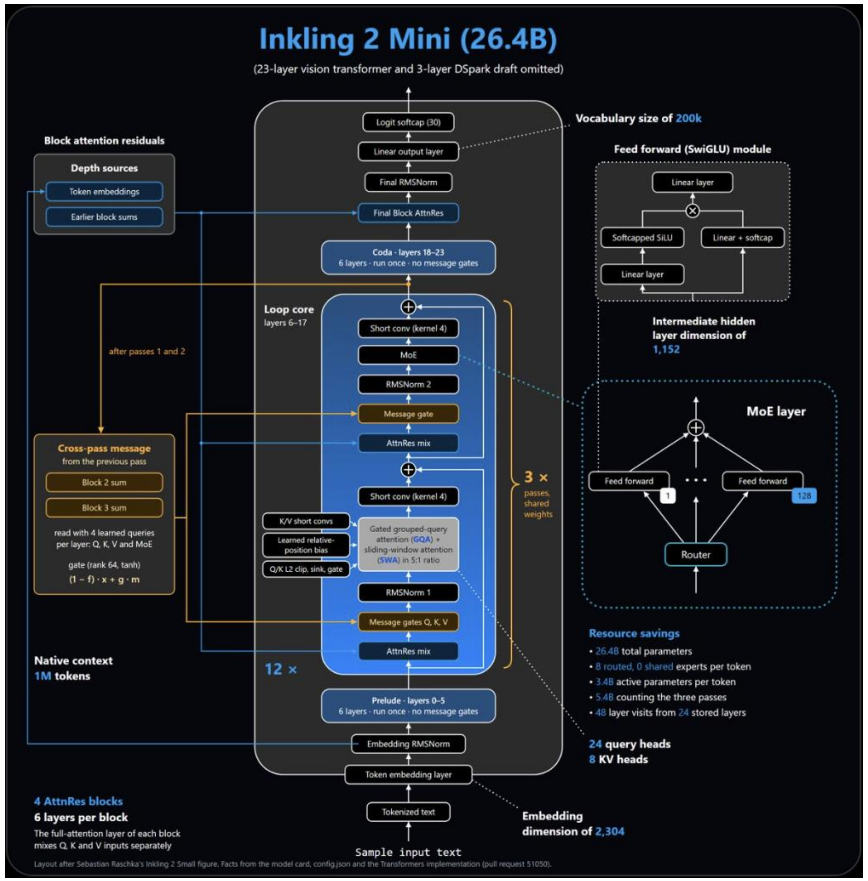
- **Latent feedback:** position t reads position t−1's hidden state. That is recurrence again, and it gives up the parallel training that let Transformers replace RNNs (Lecture 18)
- **My take:** architectures cycle; the fundamentals (BPTT, gradient flow, what parallelizes) keep paying off

Recurrence in depth: looped transformers



- **Looped transformer:** apply the same block K times. The parameters of one block, the compute of K
- **Weight sharing again:** an RNN shares weights across time; a looped transformer shares them across depth
- **Training it:** backprop through K loops of one block looks like BPTT, with the same vanishing and exploding risks; residual paths help
- **Opinion check:** if a model reasons in loops instead of written tokens, there is no chain of thought to read. Worse, or was the text never faithful anyway?

Inside a looped model: Inkling 2 Mini



- **Prelude, loop, coda:** layers 0–5 and 18–23 run once; the 12-layer core runs 3 times with shared weights. 48 layer visits from 24 stored layers
- **Two different savings:** MoE uses 8 of 128 experts per token, so 3.4B of 26.4B parameters are active. Looping reuses weights but still pays compute on every pass
- **Familiar parts:** residual connections, RMSNorm, SwiGLU feed-forward, grouped-query and sliding-window attention
- **Quick check:** counting all three passes, active parameters per token rise from 3.4B to 5.4B, but total parameters stay at 26.4B. Why?

Spend compute on parameters, or on passes?

Bigger model

- More weights store more facts and skills
- Pays full compute on every token, easy or hard
- Memory and serving cost grow with N

Loop or think longer

- Same weights, used again: Inkling 2 Mini loops its core 3 times
- Chain of thought, best-of-k: spend tokens on hard questions (Lecture 21)
- Cannot recall a fact it never stored

The trade: compute grows with the active parameters used per token ($C \approx 6ND$, Lecture 19). Looping and longer thinking buy more compute per answer without storing more weights

Ben Lengerich © University of Wisconsin-Madison 2026

Suppose an agent writes your training run…

A coding agent can now write and run your whole training pipeline. Which parts of this course would you still insist on doing, or checking, yourself?

- Which mistakes might an agent make that a loss curve would reveal?
- What would you need to see before trusting its result?
- What should a deep-learning course teach in 2030?
- **Moral question:** if the agent's model later harms someone, who is responsible: you, the agent, or its vendor?

Ben Lengerich © University of Wisconsin-Madison 2026

Who builds AI? Closed vs. open weights

Closed (API access)

- GPT, Claude, Gemini
- Usually the strongest models
- Provider controls use, updates, and safeguards
- Weights cannot be inspected or freely adapted

Open weights

- Llama, Qwen, DeepSeek, Mistral, Gemma
- Download, inspect, adapt with LoRA
- Results can be reproduced
- Once released, cannot be recalled

Recall: PEFT is what makes open weights usable by everyone (QLoRA adapts a 65B model on one GPU).

Open is a spectrum

API only	Open weights	Data and code too	Checkpoints, logs too
GPT, Claude, Gemini (query the model)	Llama, Qwen, DeepSeek (data usually withheld)	pretraining corpus, code (reproduce the recipe)	intermediate checkpoints (study training itself)

- **K2-Horizon (Nov 2026):** six fully open models, 0.9B dense to a 375B MoE with 23B active, released with the 22T-token corpus, checkpoints, and training logs
- **Why the data matters:** you can check memorization and test-set contamination, and read real training curves
- Which level of openness answers the safety argument? Which answers the science argument?

Which numbers would you trust?

Contaminated?	Optimized for?
Was the test set in the training data?	Was the model tuned to this benchmark? (Goodhart)

Comparable?	Relevant?
Same prompts, sampling, hardware, baselines?	Does it measure your task?

Ben Lengerich © University of Wisconsin-Madison 2026

Accuracy is not correctness

	Question it answers	Example
Accuracy	How often is the system right?	95% on the test set
Reliability	When can we expect it to be right?	Right on common cases, wrong on rare patients
Verifiability	Can we tell whether this answer is right?	Code with tests: errors are cheap to catch

- A 95%-accurate model can be useless, and an 80%-accurate one transformative, if its mistakes are cheap to catch.

Ben Lengerich © University of Wisconsin-Madison 2026

Training on model outputs: help or harm?

When it helps

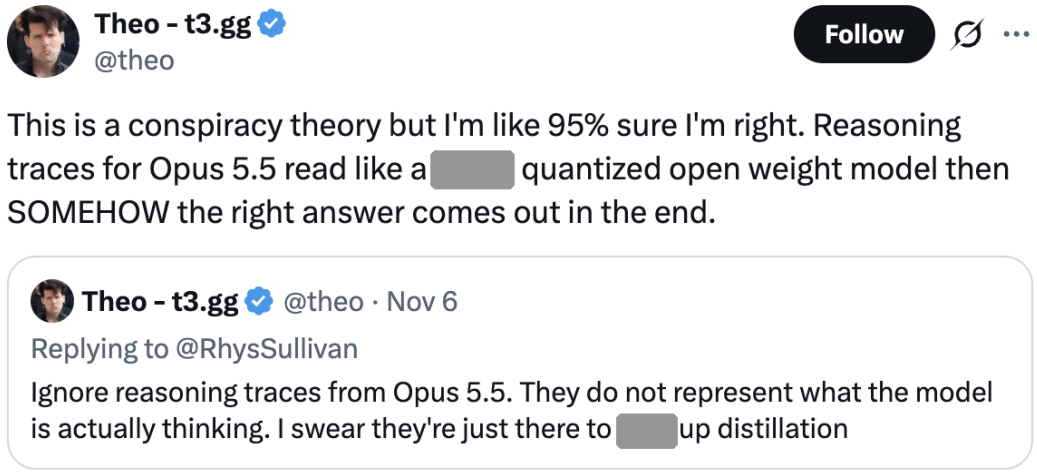
- The output can be checked: math, code with tests (RLVR, Lecture 21)
- Distillation: a strong teacher trains a small student (Lectures 20–21)
- Filling gaps: rare formats, languages, edge cases

When it can hurt

- MLE fits the corpus, not the world (Lecture 20): generated errors become targets
- Model after model trained on the last can lose the rare cases (“model collapse”)
- Once on the web, generated text is hard to filter out

- **K2-Horizon:** its data pipelines generated about 10T tokens of synthetic pretraining data
- Would you rather have more data, or more checked data?

Can you trust a reasoning trace?



- **The claim (unverified):** a closed model's visible reasoning does not match how it reaches its answers, perhaps so that distillation fails
- **Faithfulness (Lecture 21):** a written chain of thought need not be the real reason for the answer
- **Distillation (Lectures 20–21):** SFT on a stronger model's outputs. If the trace is not the real process, the student imitates the wrong thing
- **How would you test it?** Edit the trace and see if the answer changes; distill on traces vs. answers only and compare

Opinion: AI should show its work

Trust the transcript

- A chain of thought can leave out what actually drove the answer (Lecture 21)
- It disappears when the session ends
- It is tied to one model and one provider

Keep the work

- Save the rules, mappings, and checks the AI worked out
- An expert can read them, argue with them, and fix them
- They still work when you switch models

- **From my work at Intelligible:** AI analyzes a company's data and works out rules, mappings, and checks. Saved and checked, they let models reproduce all 8 known decisions in a finance pilot, every run; written instructions alone got 1 to 4
- **My take:** in high-stakes work, ask for artifacts you can check, not a transcript you have to trust

Ben Lengerich © University of Wisconsin-Madison 2026

Discuss: Accurate or intelligible?

Model A is 2% more accurate, but nobody can say why it makes any given prediction. Model B is slightly worse, but a domain expert can read and correct it. Which do you deploy?

- Does your answer change for medical triage, loan approval, or ad ranking?
- What evidence would make you trust Model A?
- Is 2% on the test set the right comparison? (Goodhart's law, Lecture 20)
- **Moral question:** does the person the model decided about have a right to an explanation they can contest?

~5 minutes with a neighbor, then share

Before you build: what would you refuse?

Two minutes, in writing: name one thing you could now build with what you learned in this course that you would refuse to build. Why? What would change your mind?

- Which lens did you use: consequences, consent, responsibility, or character?
- Does your answer change if someone else would build it anyway?
- What would you do if your employer asked you to build it?

~2 minutes writing, then share with a neighbor

Ben Lengerich © University of Wisconsin-Madison 2026

Open vs closed AI…

Should the weights of the next frontier model be released openly?

- Who gains, and who bears the risk?
- Does your answer change if the model is 10× more capable?
- Is training on a closed model's outputs (distillation) fair game?
- **Moral question:** who should get to make this decision for everyone else?

Anyone change their minds over the last few minutes?

Ben Lengerich © University of Wisconsin-Madison 2026

Project presentations: how they work

- **Tue 12/1 and Thu 12/3**, slot from the sign-up sheet
- **4 minutes** (graded window: 3–5), so every team fits
- **Every member presents:** the only individually graded part of the project
- **Upload slides to Canvas 24 hours before** your slot (PPTX, Keynote, or PDF); we present from the instructor's laptop

Suggested outline

- Title · one-sentence summary
- Problem and why it matters
- What others have done
- Your method
- Results (one key figure)
- Limitations and next steps
- Takeaway

Ben Lengerich © University of Wisconsin-Madison 2026

Final report checklist

- **Due Tue 12/8, 11:59 PM on Canvas**
- ICML template, **8 pages** plus references (start references on a new page)
- Sections: abstract, introduction, background and related work, methods, experiments, discussion (limitations, broader impact), conclusions

Experiments must include

- At least one reasonable **baseline**
- At least one **ablation or diagnostic** (remove a loss term, swap a component, use less data)
- **Error analysis:** failure cases, representative samples
- **Uncertainty** where it applies (e.g. variance over runs)

Ben Lengerich © University of Wisconsin-Madison 2026

Next Time: Project Presentations

Project Presentations: Tue 12/1 and Thu 12/3

4-minute talks; upload slides to Canvas 24 hours before your slot. No class Thursday (Thanksgiving).

Ben Lengerich © University of Wisconsin-Madison 2026

Questions?

