STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 19: Unsupervised Training of LLMs

November 12, 2026

Reading: See course homepage

From our “GPT” to today’s GPTs

From our “GPT-0” to GPT-1

- Architecture:
  - Tokenizer: characters → byte-pair encoding (BPE, 40,000 merges)
  - Normalization: pre-norm (GPT-0) vs post-norm (GPT-1, like the original Transformer)
  - Weight tying: input embedding = output layer
  - Scale (117M params):
    - Layers: 4 → 12
    - Attention heads: 4 → 12
    - Block size (max context): 8 → 512
    - Vocab: 65 characters → ~40K BPE tokens
    - Embedding dim: 64 → 768
- Training:
  - Dataset: Tiny Shakespeare (1 MB) → BooksCorpus (~7,000 unpublished books)
  - Learning rate: constant → 2,000-step warmup + cosine decay

Ben Lengerich © University of Wisconsin-Madison 2026

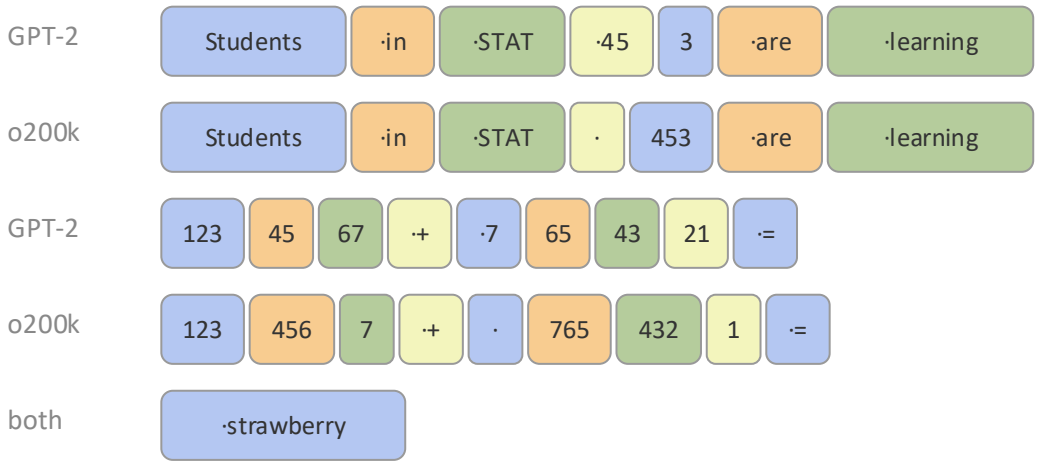
Tokenization: byte-pair encoding (BPE)

- Characters: tiny vocab, very long sequences. Words: huge vocab, unknown words
- BPE: start from characters (GPT-2: bytes); repeatedly **merge the most frequent adjacent pair** into a new token

Step	Merge	Sequence	Length
0		aaabdaaabac	11
1	aa → Z	ZabdZabac	9
2	ab → Y	ZYdZYac	7
3	ZY → X	XdXac	5

- Vocab size: GPT-1 ~40K, GPT-2/3 50,257, Llama 3 128K, gpt-oss 201K

Real tokenizations (tiktoken):



- · marks a space; spaces are part of tokens
- Cost, speed, and context length are all counted in tokens
- The model never sees the letters inside “strawberry”, and digits are chunked arbitrarily

Ben Lengerich © University of Wisconsin-Madison 2026

From GPT-1 to GPT-2

GPT-2: [Radford et al., Language Models are Unsupervised Multitask Learners]

- Architecture:
  - LayerNorm moved to the input of each sub-block (pre-norm) + a final LayerNorm
  - Byte-level BPE: any string can be encoded, no unknown tokens
  - Scale: 4 sizes (124M → 1.5B params); biggest:
    - Layers: 12 → 48
    - Attention heads: 12 → 25
    - Embedding dim: 768 → 1600
    - Block size (max context): 512 → 1024
    - Vocab: ~40K → 50,257 tokens
- Training:
  - Dataset: BooksCorpus → WebText (~40 GB, ~8M web pages linked from Reddit)
  - No fine-tuning: tasks posed as text (“zero-shot”)

You can reproduce GPT-2 (124M) yourself:
https://github.com/karpathy/nanoGPT
(takes 4 days to train on an 8xA100 machine)

From GPT-2 to GPT-3

GPT-3: [Brown et al., Language Models are Few-Shot Learners]

- Architecture:
  - Same design as GPT-2, plus alternating dense and locally banded sparse attention layers
  - Scale (1.5B → 175B params):
    - Block size (max context): 1024 → 2048
    - Layers: 48 → 96
    - Embedding dim: 1600 → 12,288
    - Attention heads: 25 → 96 (each of dim 128)
- Training:
  - Dataset: WebText → filtered Common Crawl (~570 GB) + WebText2, books, Wikipedia
  - ~300B training tokens
  - Headline result: few-shot “in-context learning” from examples in the prompt, with no weight updates (later today)

From GPT-3 to GPT-4: the curtain closes

- What OpenAI disclosed (GPT-4 Technical Report, 2023):
  - Transformer-based, pretrained to predict the next token, then fine-tuned with RLHF
  - Accepts images and text as input
  - “No further details about the architecture (including model size), hardware, training compute, dataset construction, training method”
- What is not known:
  - Online parameter counts and “mixture of experts” claims are rumors, not published facts
- Next: models whose weights and reports are public

Ben Lengerich © University of Wisconsin-Madison 2026

Open-weight models: we can see inside again

	GPT-2 XL (2019)	GPT-3 (2020)	Llama 3.1 405B (2024)	DeepSeek-V3 (2024)	gpt-oss-120b (2025)
Params: total / active	1.5B / 1.5B	175B / 175B	405B / 405B	671B / 37B	117B / 5.1B
Layers × width	48 × 1,600	96 × 12,288	126 × 16,384	61 × 7,168	36 × 2,880
Attention	MHA, 25 heads	MHA + banded sparse	GQA: 128 q / 8 kv heads	MLA (compressed KV)	GQA: 64 q / 8 kv; alternating 128-token window / full
Position	learned	learned	RoPE	RoPE	RoPE + YaRN
Norm · MLP	LayerNorm · GELU	LayerNorm · GELU	RMSNorm · SwiGLU	RMSNorm · SwiGLU MoE	RMSNorm · SwiGLU MoE
Experts	–	–	– (dense)	1 shared + 256 routed, top-8	128, top-4
Context	1K	2K	128K	128K	128K (4K, extended)
Vocab	50K	50K	128K	129K	201K
Training tokens	40 GB of text	300B	15.6T	14.8T	not disclosed

The modern decoder recipe (vs. GPT-2)

Part	GPT-2 (2019)	Typical in 2025–26	Why
Normalization	LayerNorm	RMSNorm	cheaper, just as stable
MLP	GELU	SwiGLU (gated)	better quality per parameter
Positions	learned absolute	RoPE	learned table ends at training length; RoPE is relative
KV heads	one per query head	GQA / MLA	smaller KV cache, faster generation
FFN	dense	Mixture of Experts	more parameters, same compute per token
Context	1K	128K+	RoPE scaling, sliding windows
Vocab	50K	128K–200K	fewer tokens per text, more languages

$$\mathrm{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d}\sum_i x_i^2 + \epsilon}} \odot \mathbf{g}$$

$$\mathrm{SwiGLU}(\mathbf{x}) = \big(\mathrm{SiLU}(\mathbf{x}W_1) \odot \mathbf{x}W_3\big)W_2$$

Unchanged since 2018:

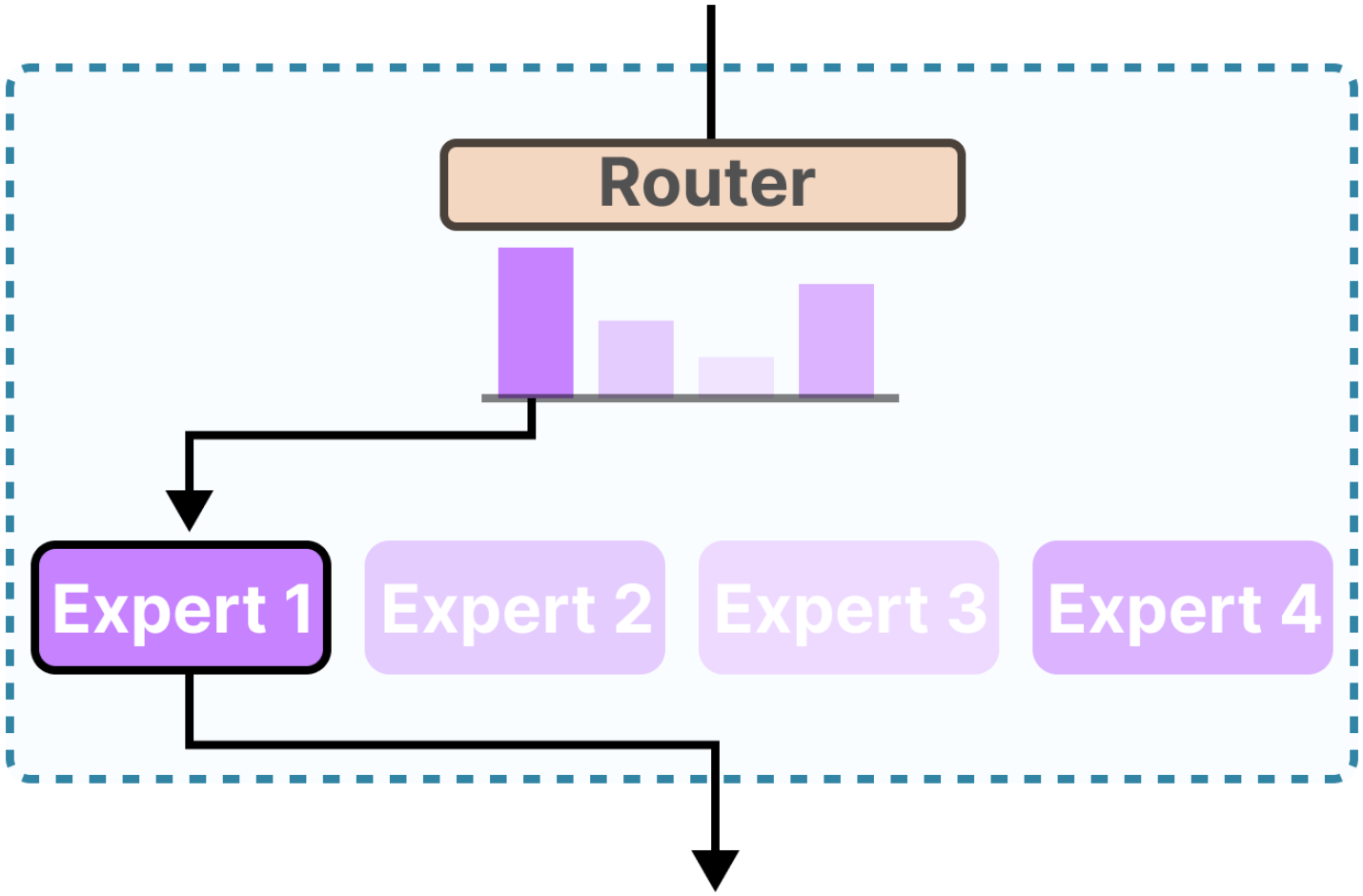
decoder-only

causal self-attention + MLP + residual stream

next-token cross-entropy pretraining

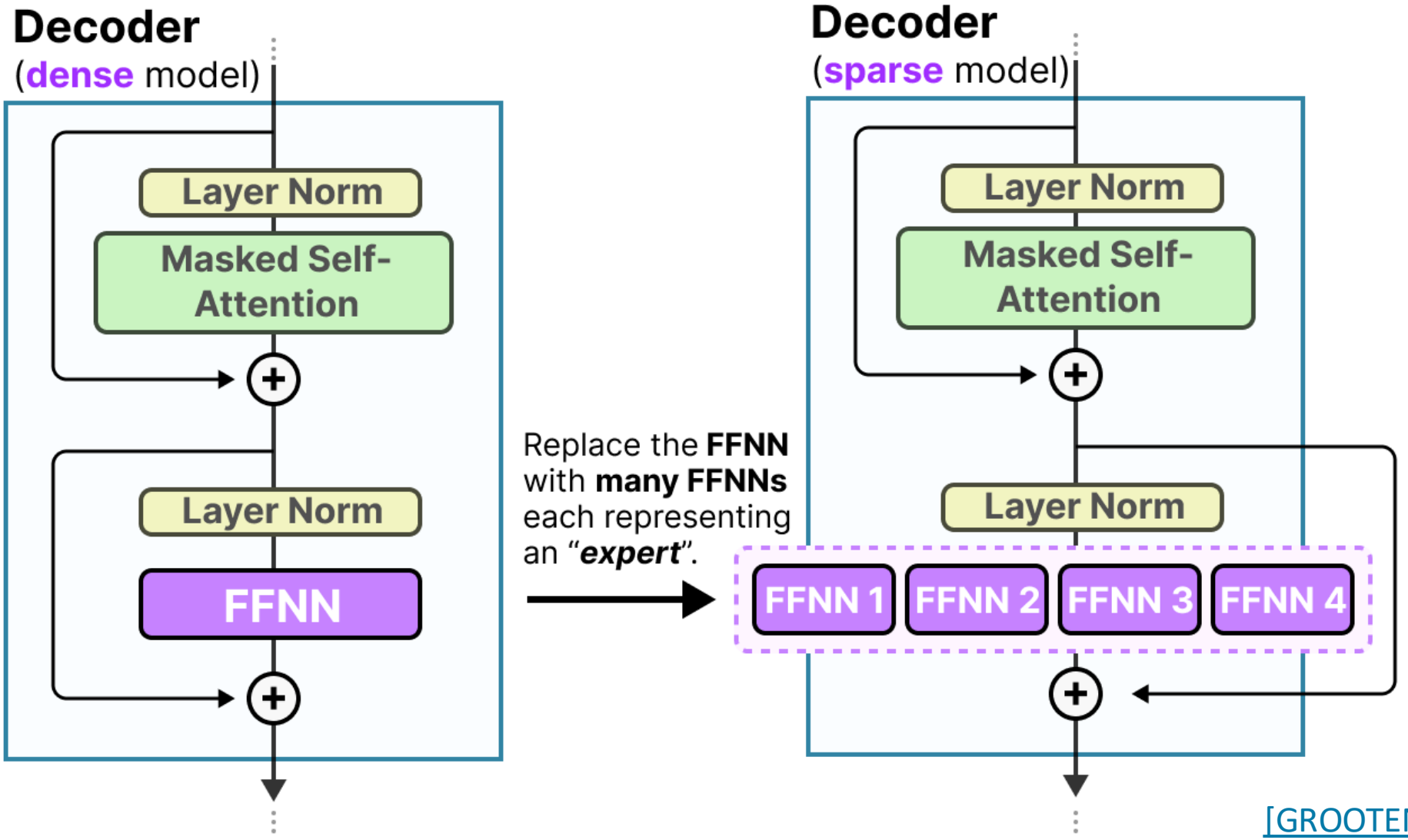
Mixture of Experts

Mixture of Experts



[GROOTENDORST]

Mixture of Experts inside Transformer Decoder



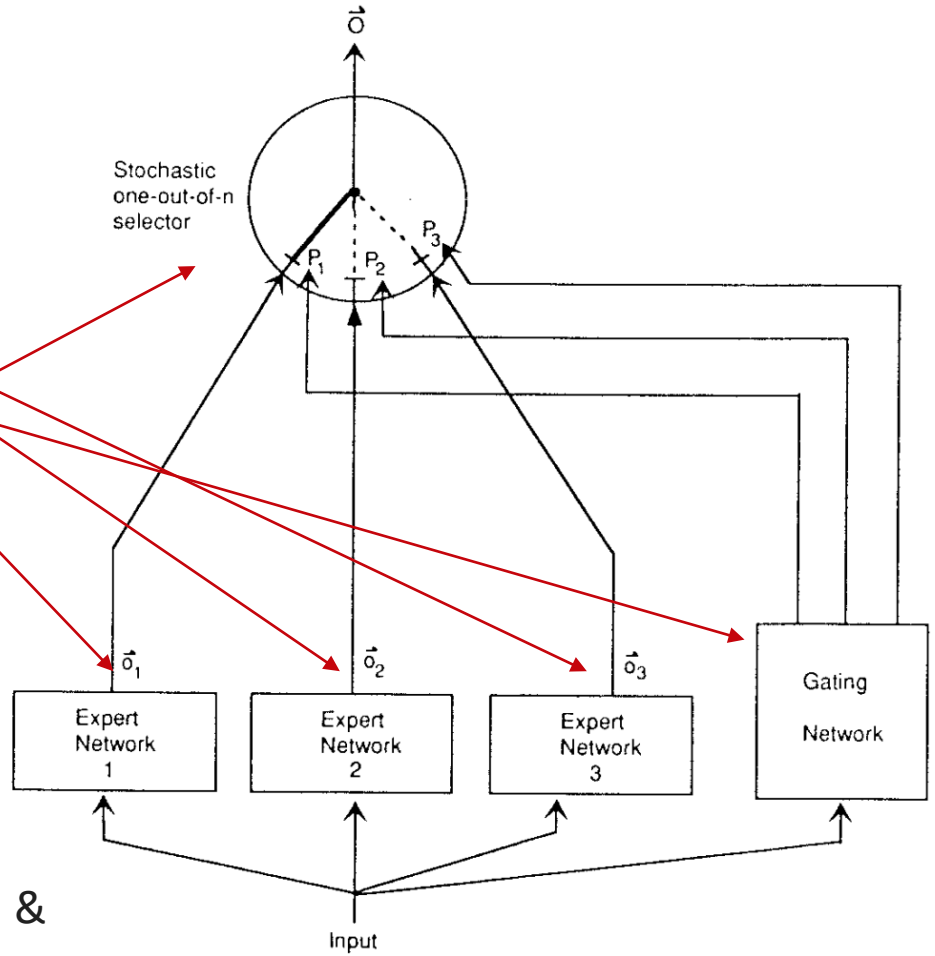
[GROOTENDORST]

Mixture of Experts: A probabilistic idea

- Let

$$P(Y \mid X) = \sum_m g_m(X) \cdot P_m(Y \mid X)$$

- Constrain $\sum_m g_m(X) = 1$ and $g_m(X) \geq 0 \; \forall \; m, \mathrm{X}.$



"Adaptive Mixtures of Local Experts" Jacobs et al 1991

How would you estimate these parameters?

[Hierarchical mixtures of experts and the EM algorithm; Jordan & Jacobs 1994]

Ben Lengerich © University of Wisconsin-Madison 2026

Mixture of Experts: Unifies Several Approaches

Let

$$P(Y \mid X) = \sum_m g_m(X) \cdot P_m(Y \mid X)$$

- **Mixture of Experts** [Jacobs et al 1991]: $g_m(X)$ is a learned gating function.
- **Bagging** [Breiman 1996]: $g_m(X) = \frac{1}{M}$ is a constant, uniform weighting.
- **Boosting** [Freund & Schapire 1997]: $g_m(X) = \alpha_m$ is a constant per-expert weight.

Ben Lengerich © University of Wisconsin-Madison 2026

Bonus

Mixture of Experts: Some analysis

Let

$$P(Y \mid X) = \sum_m g_m(X) \cdot P_m(Y \mid X)$$

- Let's examine the mean functions. Define:

$$\bar{f}(x) \coloneqq \mathrm{E}[\, Y \mid X \,] = \sum_m g_m(X) E_m[\, Y \mid X \,] = \sum_m g_m(X)\, f_m(x)$$

- Let's compare:
  - $\epsilon(x) \coloneqq \left(Y - \bar{f}(x)\right)^2$ ← "Ensemble error"
  - $\bar{\epsilon}(x) \coloneqq \frac{1}{M}\sum_m \left(Y - f_m(x)\right)^2$ ← "Average expert error"

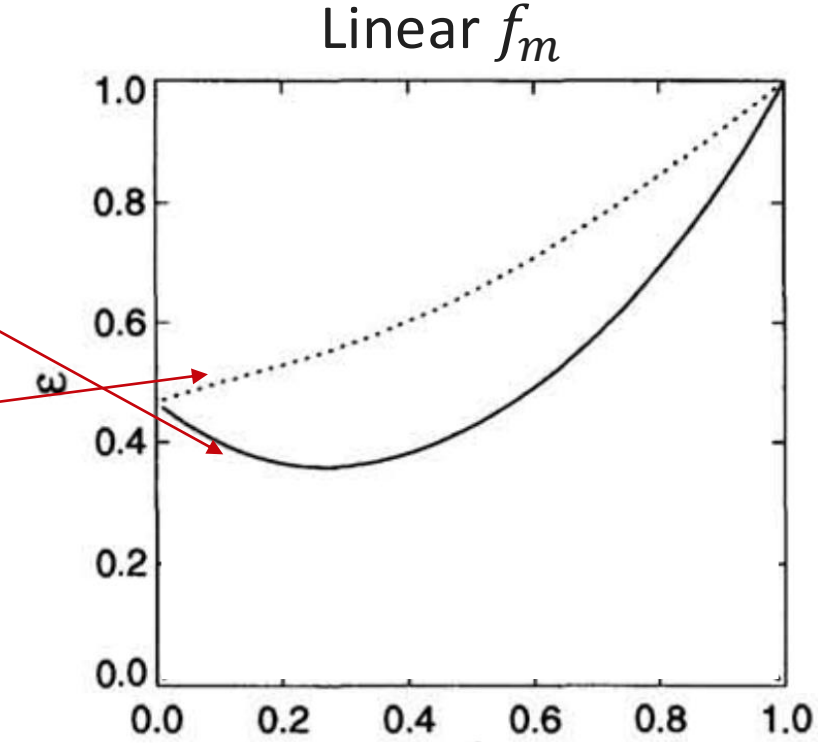
Will these be minimized for the same ensemble?

Ben Lengerich © University of Wisconsin-Madison 2026

Bonus

Mixture of Experts: Some analysis

- $\epsilon(x) := \left(Y - \bar{f}(x)\right)^2$ ← "Ensemble error"
- $\bar{\epsilon}(x) := \frac{1}{M}\sum_m \left(Y - f_m(x)\right)^2$ ← "Average expert error"



[Sollich & Krogh 1995]

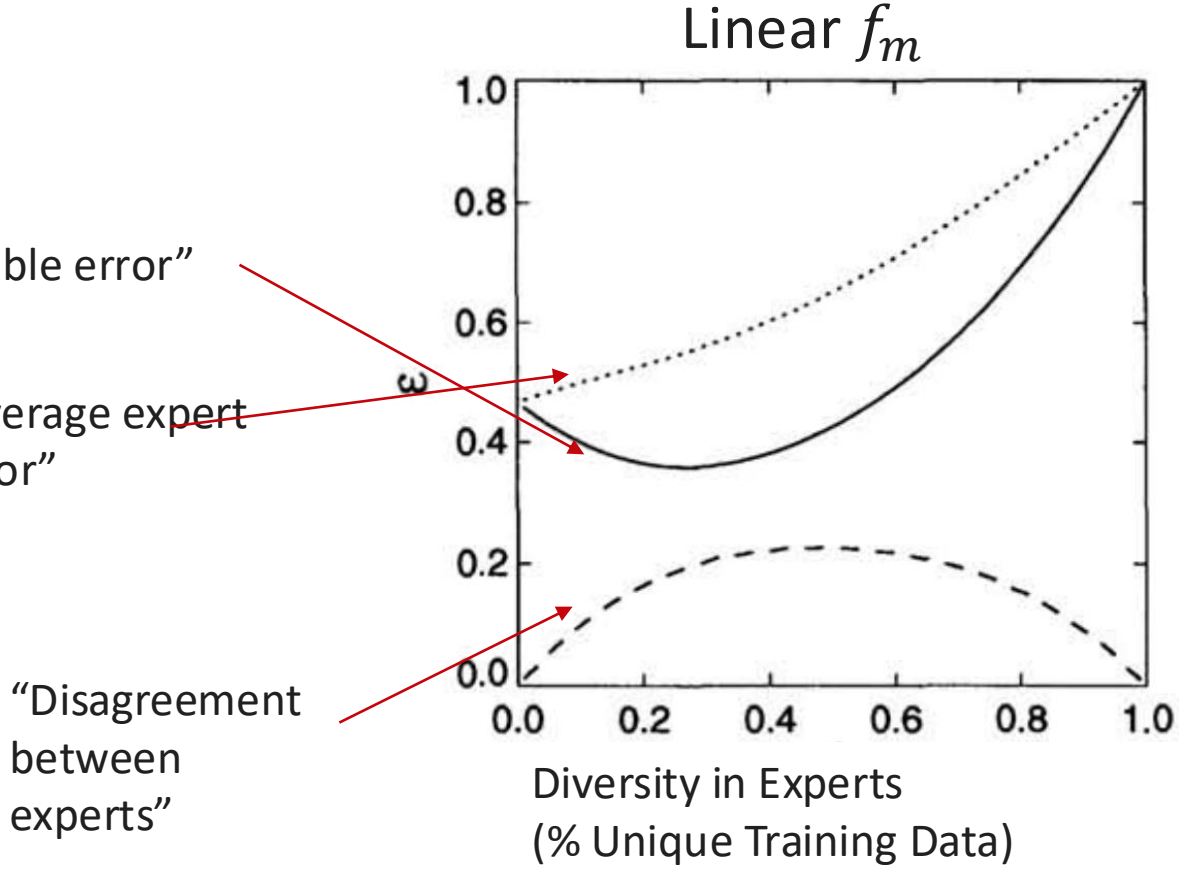
Bonus

Mixture of Experts: Some analysis

- $\epsilon(x) := \left(Y - \bar{f}(x)\right)^2$ ← "Ensemble error"
- $\bar{\epsilon}(x) := \frac{1}{M}\sum_m \left(Y - f_m(x)\right)^2$ ← "Average expert error"

Intuition: Expert errors are wrong in individualized ways and cancel out through consensus.

→ Slight "overfitting" of experts helps!



[Sollich & Krogh 1995]

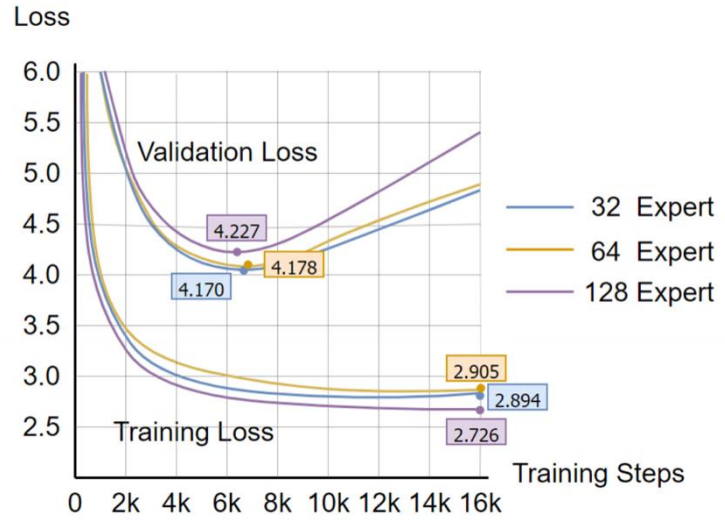
Ben Lengerich © University of Wisconsin-Madison 2026

Mixture of Experts: In LLMs

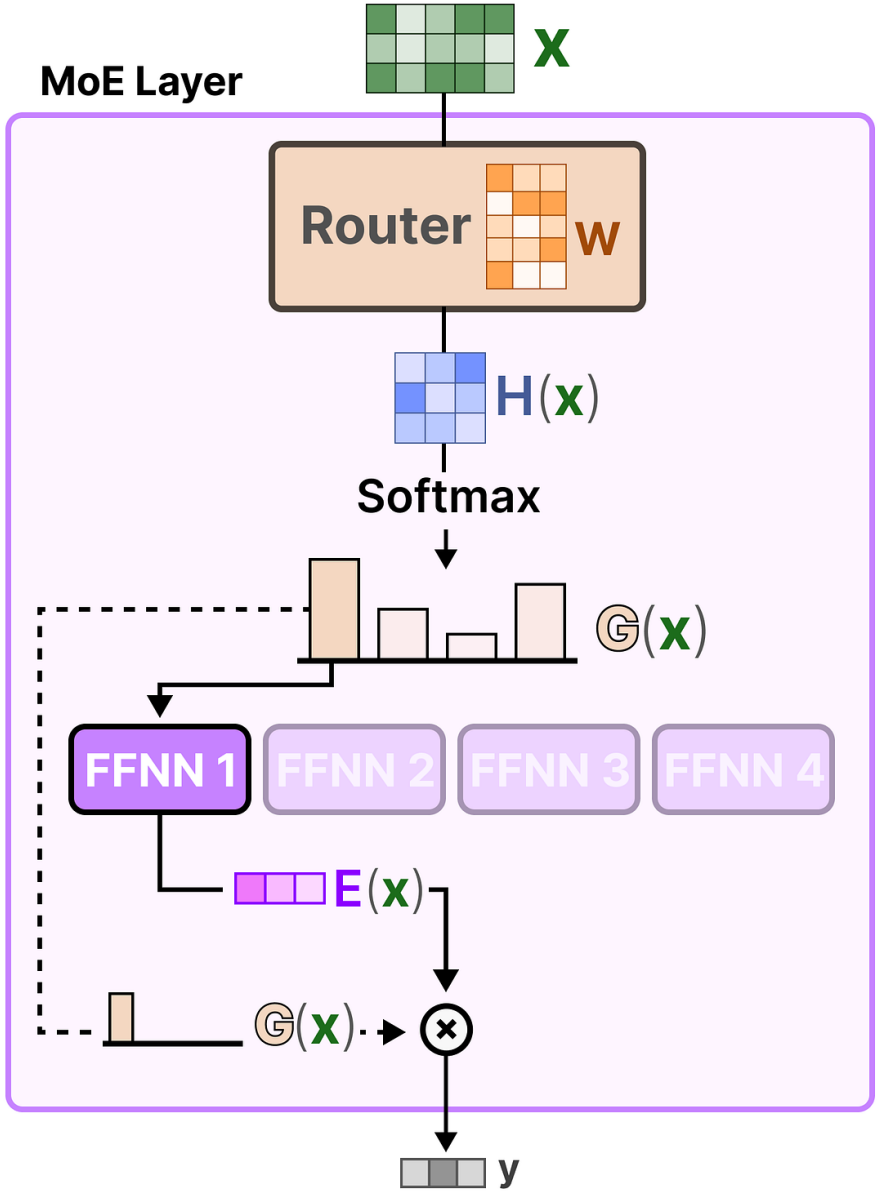
- Implications for serving?

- Implications for training?

Easy to have experts over-specialize



[XIE ET AL 2022]



[GROOTENDORST]

Mixture of Experts in today's LLMs

$$\mathbf{g}(\mathbf{x}) = \mathrm{softmax}\big(\mathrm{TopK}(\mathbf{x}W_r,\, k)\big), \qquad \mathbf{y} = \sum_{i \in \mathrm{TopK}} g_i(\mathbf{x})\, E_i(\mathbf{x})$$

Model	Experts / layer	Used per token	Total	Active
Mixtral 8x7B (2024)	8	2	47B	13B
DeepSeek-V3 (2024)	256 + 1 shared	8 + 1 shared	671B	37B
gpt-oss-120b (2025)	128	4	117B	5.1B

- Same gate as 1991, but **sparse**: only the top-k experts run
- Capacity grows with **total** params; cost per token with **active** params
- Training: keep the router balanced, or a few experts get every token
- Serving: every expert must sit in GPU memory

Summary: From Transformer to GPT

Component	Transformer	GPT
Architecture	Encoder-decoder (full)	Decoder-only
Attention	Full self-attention	Masked (causal) self-attention
Positional encoding	Sinusoidal (original)	Learned (GPT-1 to 3); RoPE in most models today
Output	Task-specific	Next-token prediction
Training objective	Flexible (e.g., translation)	Next-token maximum likelihood (cross-entropy)
Inference	Depends on task	One token at a time: greedy, temperature, top-k / top-p; KV cache

Ben Lengerich © University of Wisconsin-Madison 2026

Summary: From GPT-1 to today

- **Same skeleton: decoder-only, causal self-attention + MLP + residuals, next-token prediction**
- **Architecture:**
  - Scale: 117M (GPT-1) → 175B dense (GPT-3) → hundreds of billions total with MoE
    - Block size (max context): 512 → 128K+
    - Layers: 12 → 36–126; width 768 → 2.9K–16K
    - Vocab: ~40K → 128K–200K tokens
  - Parts swapped in: RMSNorm, SwiGLU, RoPE, GQA / MLA, Mixture-of-Experts
- **Training:**
  - Data: BooksCorpus → ~15 trillion tokens (today)
  - Then post-training: SFT + RL from human feedback (next week)

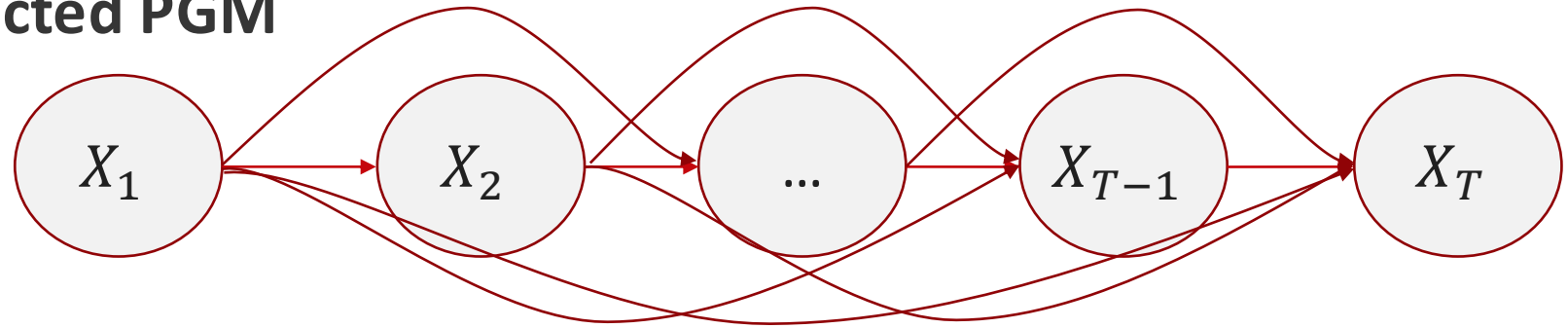
Today: Unsupervised Training of LLMs

- **Same skeleton: decoder-only, causal self-attention + MLP + residuals**
- **Architecture:**
  - Scale: 117M (GPT-1) → 175B dense (GPT-3) → hundreds of billions total with MoE
  - Block size 512 → 128K+; vocab ~40K → 128K–200K
  - Parts swapped in: RMSNorm, SwiGLU, RoPE, GQA / MLA, Mixture-of-Experts
- **Training:**
  - **Data: BooksCorpus → ~15 trillion tokens**
  - Then post-training: SFT + RL from human feedback (next week)

Unsupervised Training of LLMs

Recall GPT training objective: MLE

- **Directed PGM**



$$P_\theta(X) = \prod_i \prod_t P_\theta(X_{i,t} \mid X_{i,<t})$$

- **Probabilistic objective:** Max log-likelihood of observed seqs

$$\max_\theta \sum_i \sum_t \log P_\theta\left(X_{i,t} \mid X_{i,<t}\right)$$

[Radford et al., Improving Language Understanding by Generative Pre-Training]

The pretraining loop: every token is a label



- Each block of T tokens gives **T prediction problems**, with no human labels ("self-supervised")
- AdamW, learning-rate warmup then decay, mixed precision (bf16 / FP8)
- One pass (or a few) over the data: training loss ≈ test loss
- **Perplexity**: effective number of equally likely choices per token

$$\mathrm{PPL} = \exp\Big(-\frac{1}{T} \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}) \Big)$$

- Example: p(the) = ¼, p(dog | the) = ¼ → p = 1/16
- NLL = $-\log_2(1/16)$ = 4 bits → 2 bits/token → PPL = 2^2 = 4
- Same PPL in nats: the base cancels
- Random init: PPL ≈ V

We've had MLE-based Language Models for a while…

Large Language Models in Machine Translation 2007

Thorsten Brants **Ashok C. Popat** **Peng Xu** **Franz J. Och** **Jeffrey Dean**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com



Some fun:
https://github.com/LRitzdorf/TheJeffDeanFacts

We've had MLE-based Language Models for a while…

Large Language Models in Machine Translation 2007

ɔk C. Popat Peng Xu Franz J. Och Jeffrey Dean

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
popat,xp,och,jeff}@google.com

This paper reports on the benefits of large-scale statistical language modeling in machine translation. A distributed infrastructure is proposed which we use to train on up to 2 trillion tokens, resulting in language models having up to 300 billion n-grams. It is capable of providing smoothed probabilities for fast, single-pass decoding. We introduce a new smoothing method, dubbed *Stupid Backoff*, that is inexpensive to train on large data sets and approaches the quality of Kneser-Ney Smoothing as the amount of training data increases.

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{L} \hat{P}(w_i|w_{i-n+1}^{i-1})$$

We've had MLE-based Language Models for a while…

Large Language Models in Machine Translation 2007

Thorsten Brants **Ashok C. Popat** **Peng Xu** **Franz J. Och** **Jeffrey Dean**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com

- Why wasn't this the LLM moment?
  - Modeled n-grams, not *relationships* of token embeddings
  - Transformer architecture
  - Scale, GPU acceleration

MLE → Emergent Understanding?

- ”The simplest way to predict the next token is to understand what happened throughout the context.”
- To predict the word "*is*" in "*The capital of France ___ Paris.*", the model must:
  - Resolve subject-verb agreement
  - Recognize a factual structure
  - Know the topic is geography

Ben Lengerich © University of Wisconsin-Madison 2026

Scale & Emergent Capabilities

What happens as we scale training?

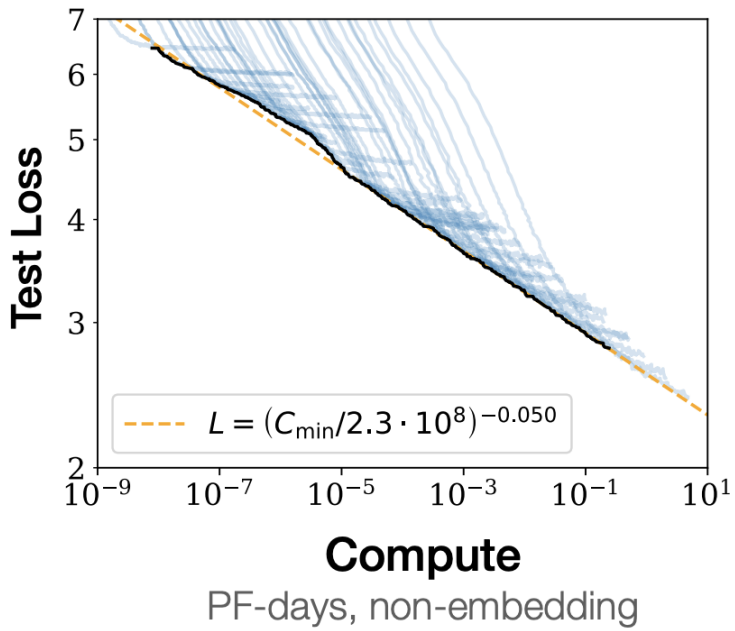


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

"Scaling Laws for Neural Language Models". Kaplan et al. 2020

What happens as we scale training?

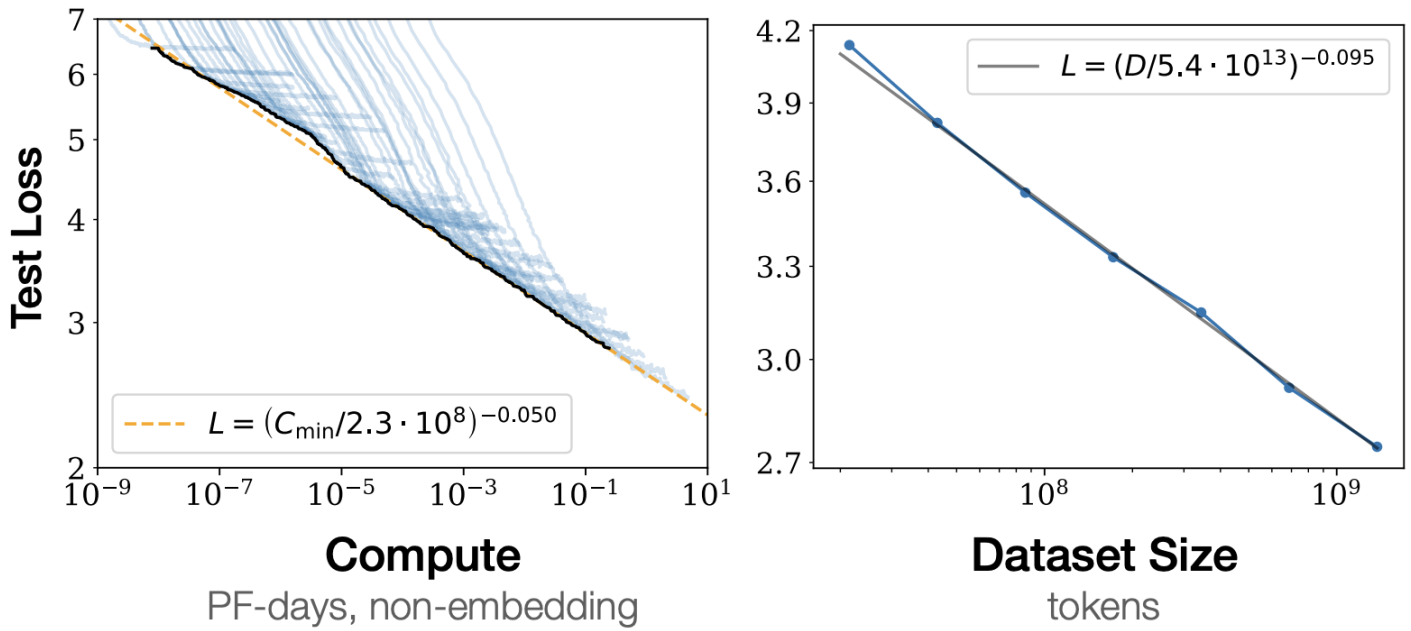


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

"Scaling Laws for Neural Language Models". Kaplan et al. 2020

What happens as we scale training?

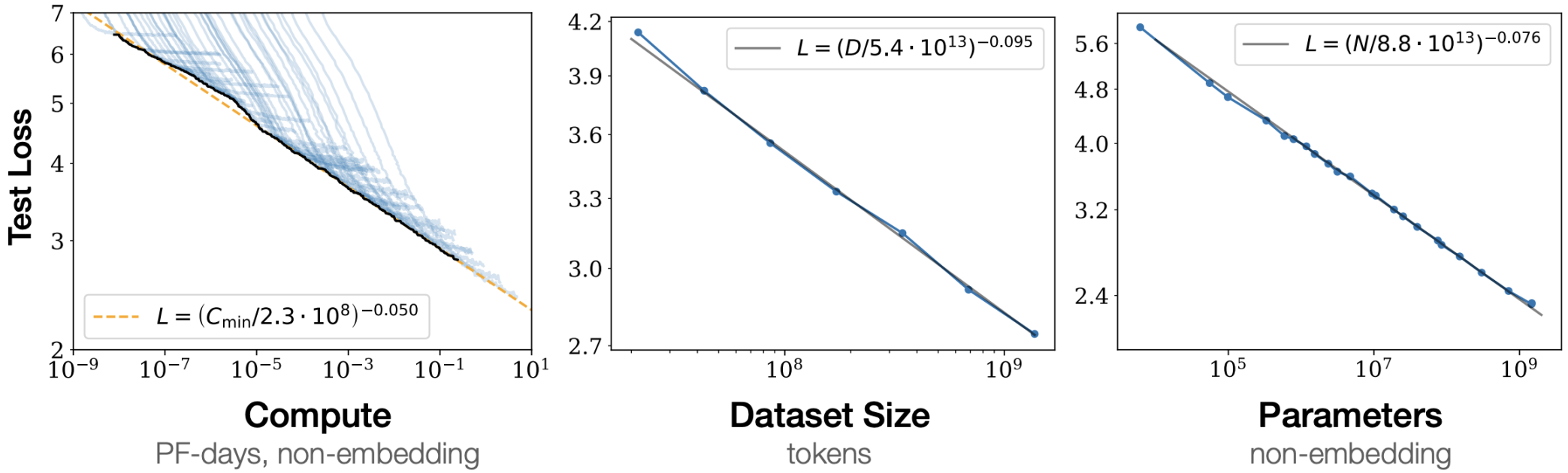
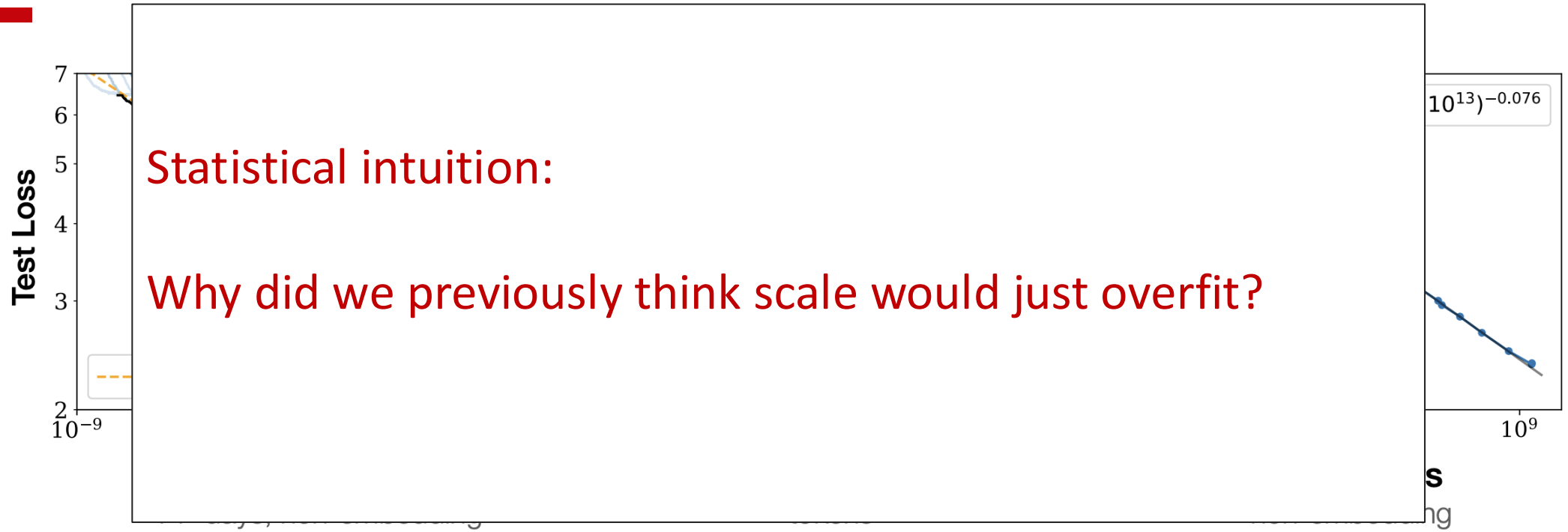


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

"Scaling Laws for Neural Language Models". Kaplan et al. 2020

What happens as we scale training?



Statistical intuition:

Why did we previously think scale would just overfit?

Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

"Scaling Laws for Neural Language Models". Kaplan et al. 2020

Counting compute: C ≈ 6ND

$$C \approx \underbrace{6}_{2\ \text{fwd}+4\ \text{bwd}} \times \underbrace{N}_{\text{params}} \times \underbrace{D}_{\text{tokens}} \quad \text{FLOPs}$$

Model	N (active)	D (tokens)	C ≈ 6ND
GPT-3 (2020)	175B	300B	3.15×10^{23}
Chinchilla (2022)	70B	1.4T	5.9×10^{23}
DeepSeek-V3 (2024)	37B	14.8T	3.3×10^{24}
Llama 3.1 405B (2024)	405B	15.6T	3.8×10^{25}

- MoE: N counts **active** parameters only
- DeepSeek-V3 reported 2.66M H800 GPU-hours of pretraining → ≈ 340 TFLOP/s sustained per GPU

Compute-optimal scaling: Chinchilla

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \qquad \text{minimize subject to } 6ND = C$$

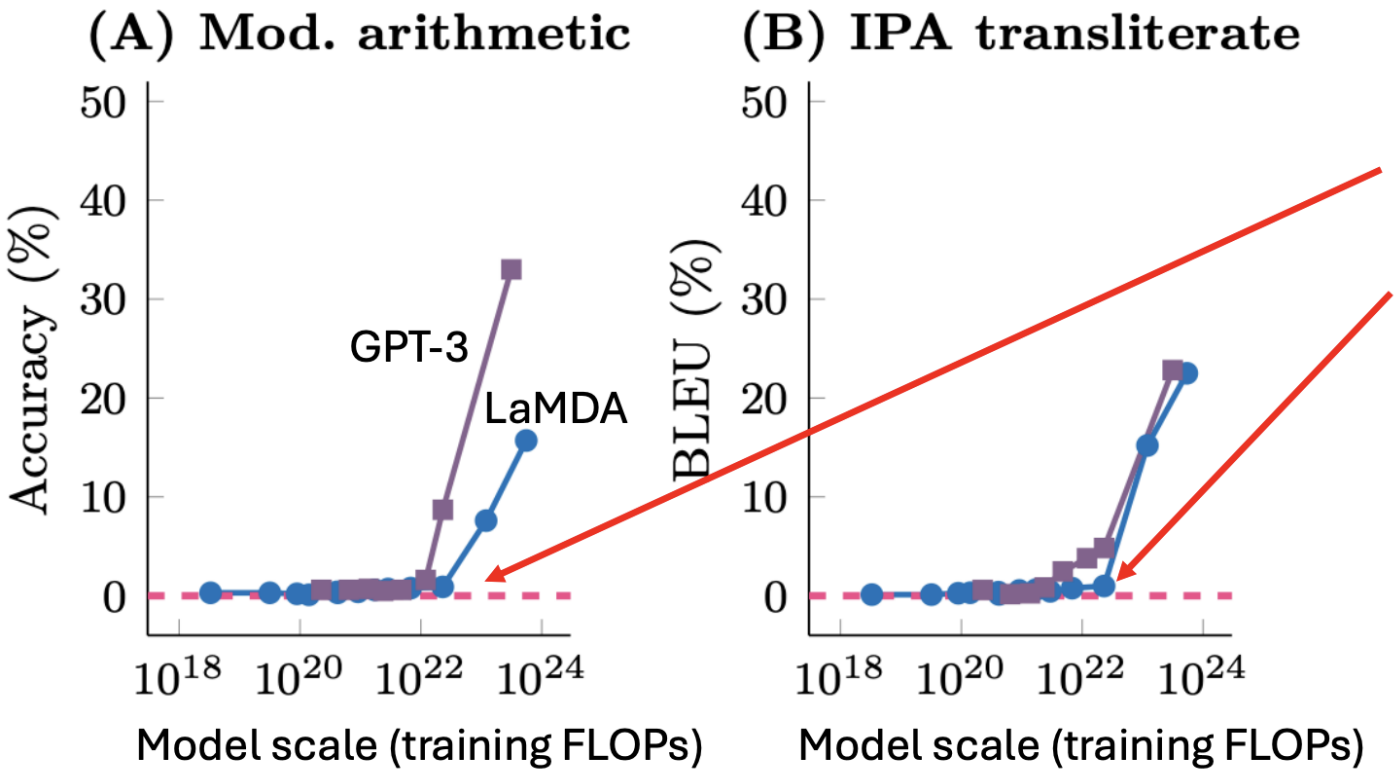
$$\Rightarrow \; N_{\text{opt}} \propto C^{a}, \;\; D_{\text{opt}} \propto C^{b}, \qquad a = \frac{\beta}{\alpha + \beta} \approx 0.5 \approx b$$

- Double the compute → grow **both** N and D by √2 (Kaplan 2020: mostly grow N, $N \propto C^{0.73}$)
- Rule of thumb: **≈ 20 training tokens per parameter**
- Chinchilla (70B, 1.4T tokens) beat Gopher (280B, 300B tokens) at equal compute
- DeepSeek LLM (reading): same analysis with M = non-embedding FLOPs/token (a = 0.52, b = 0.48), plus power laws for the best batch size and learning rate

Beyond compute-optimal: train small models longer

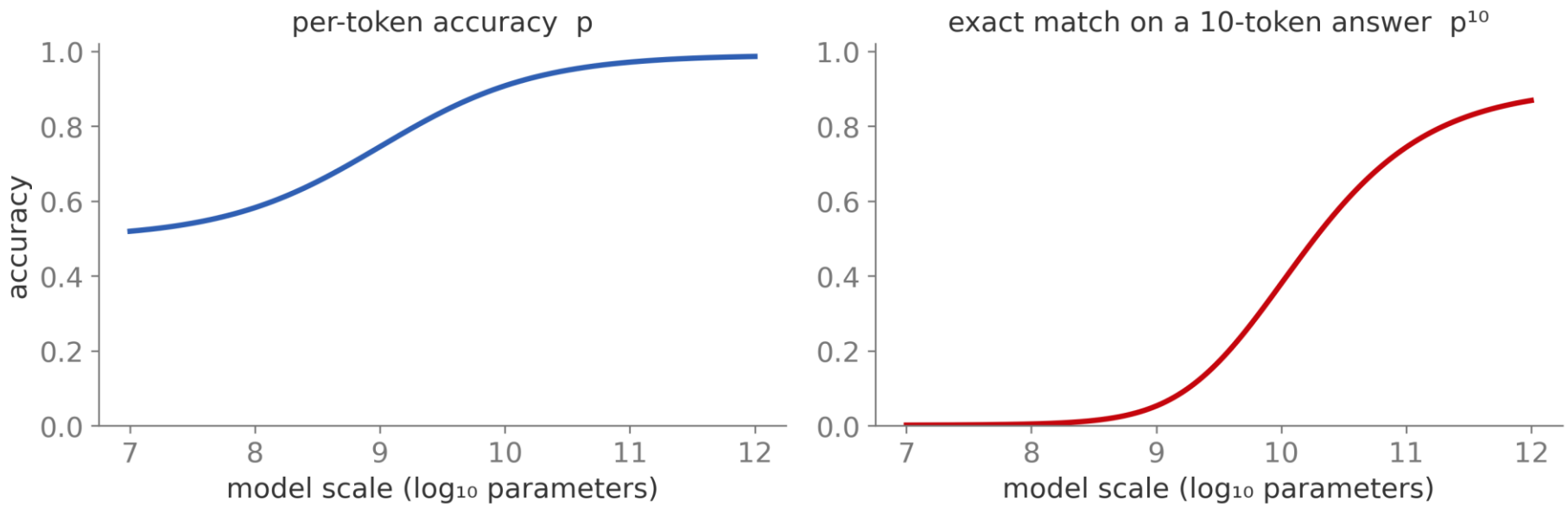
- Chinchilla minimizes **training** cost; serving cost grows with N × usage
  - Llama 3 8B: Chinchilla-optimal ≈ 200B tokens; trained on **15T** (≈ 1,900 tokens/param) and still improving
  - Smaller model + more data = cheaper to serve at the same quality
- Next limit: **data**
  - Public human-written text: ≈ 300T tokens (90% CI: 100T–1,000T); could be used up between 2026 and 2032 (Epoch AI)
  - Repeating data up to ~4 epochs ≈ as good as fresh data (Muennighoff et al. 2023)
  - Responses: better filtering, synthetic data, code / math, other modalities

Smooth improvements → sharp emergent ability?



An ability is emergent if it is not present in smaller models but is present in larger models [Wei, et al (2022). Emergent Abilities of Large Language Models

Emergence, or a choice of metric?



- Toy model: per-token accuracy p rises smoothly; exact match on a 10-token answer is p^{10}, a **sudden jump**
- Schaeffer et al. 2023: with continuous metrics, many “emergent” abilities improve smoothly and predictably

Ben Lengerich © University of Wisconsin-Madison 2026

Example of emergence: In-Context Learning

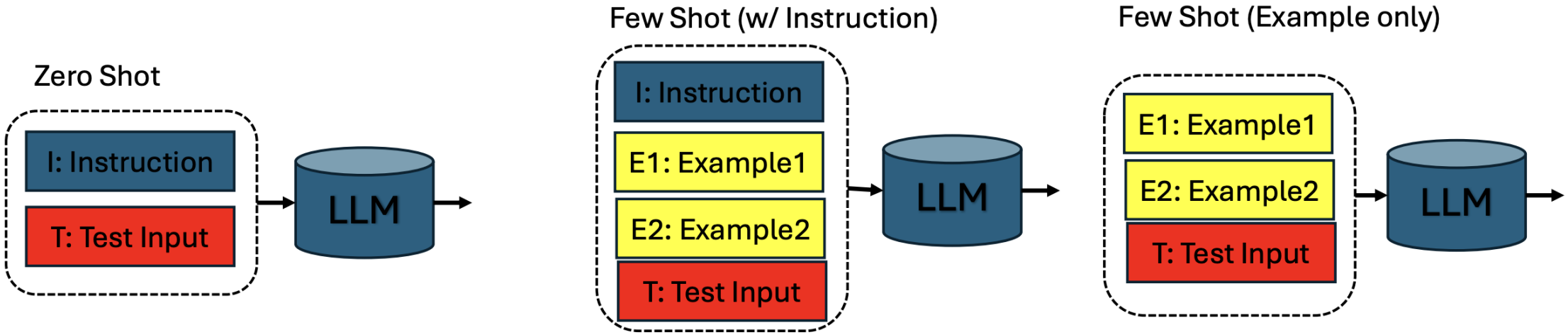
I: Instruction — Translate English to French

E1: Example1 — [en]: A discomfort which lasts. [fr]: Un malaise qui dure

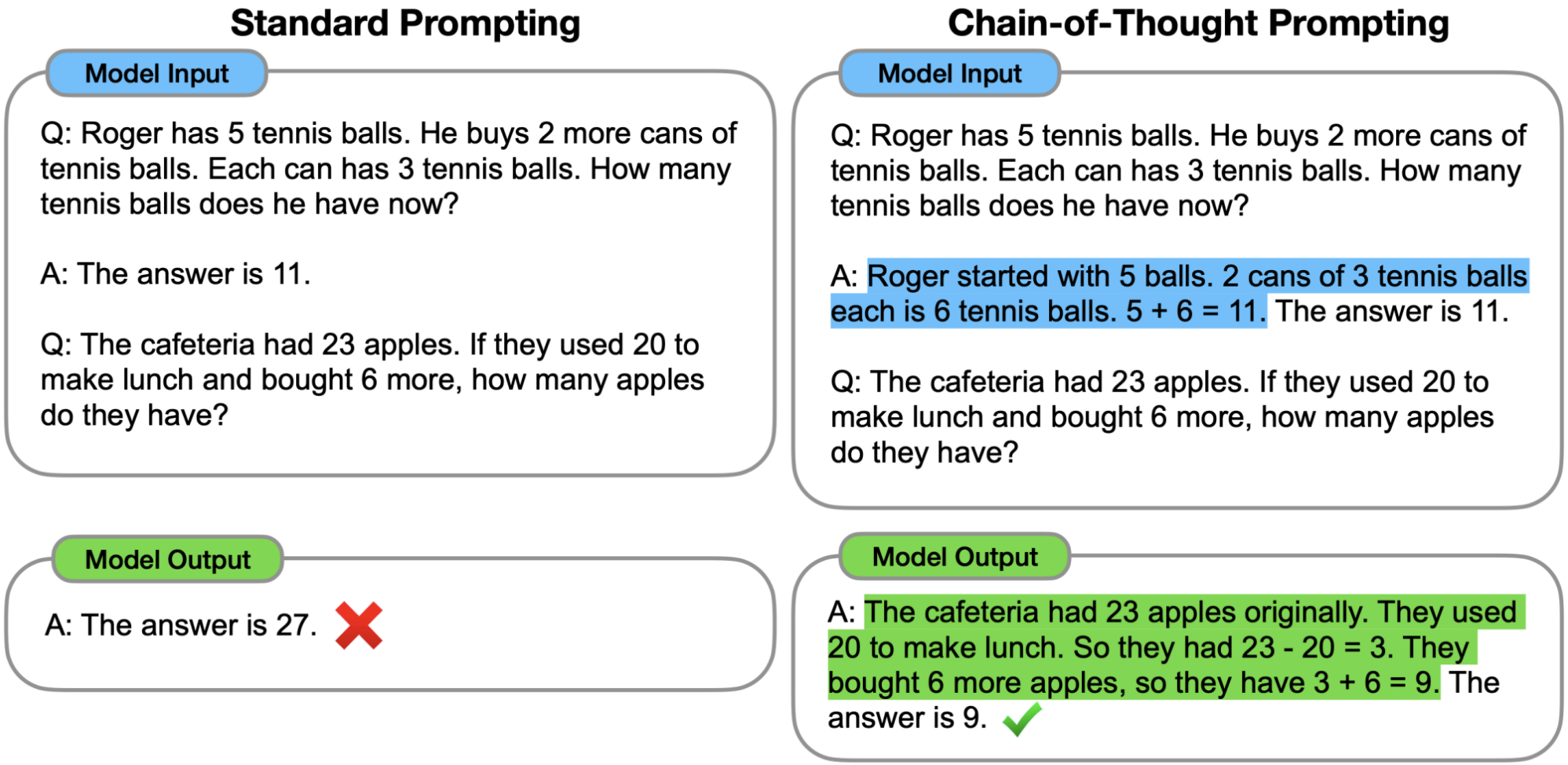
E2: Example2 — [en]: HTML is a language for formatting formatage [fr]: HTML est un langage de

T: Test Input — [en]: After you become comfortable with formatting [fr]:

No weights change: the "learning" happens in one forward pass



Example of emergence: Chain-of-Thought



Wei, et al. (2023) Chain-of-Though Prompting Elicits Reasoning in LLMs

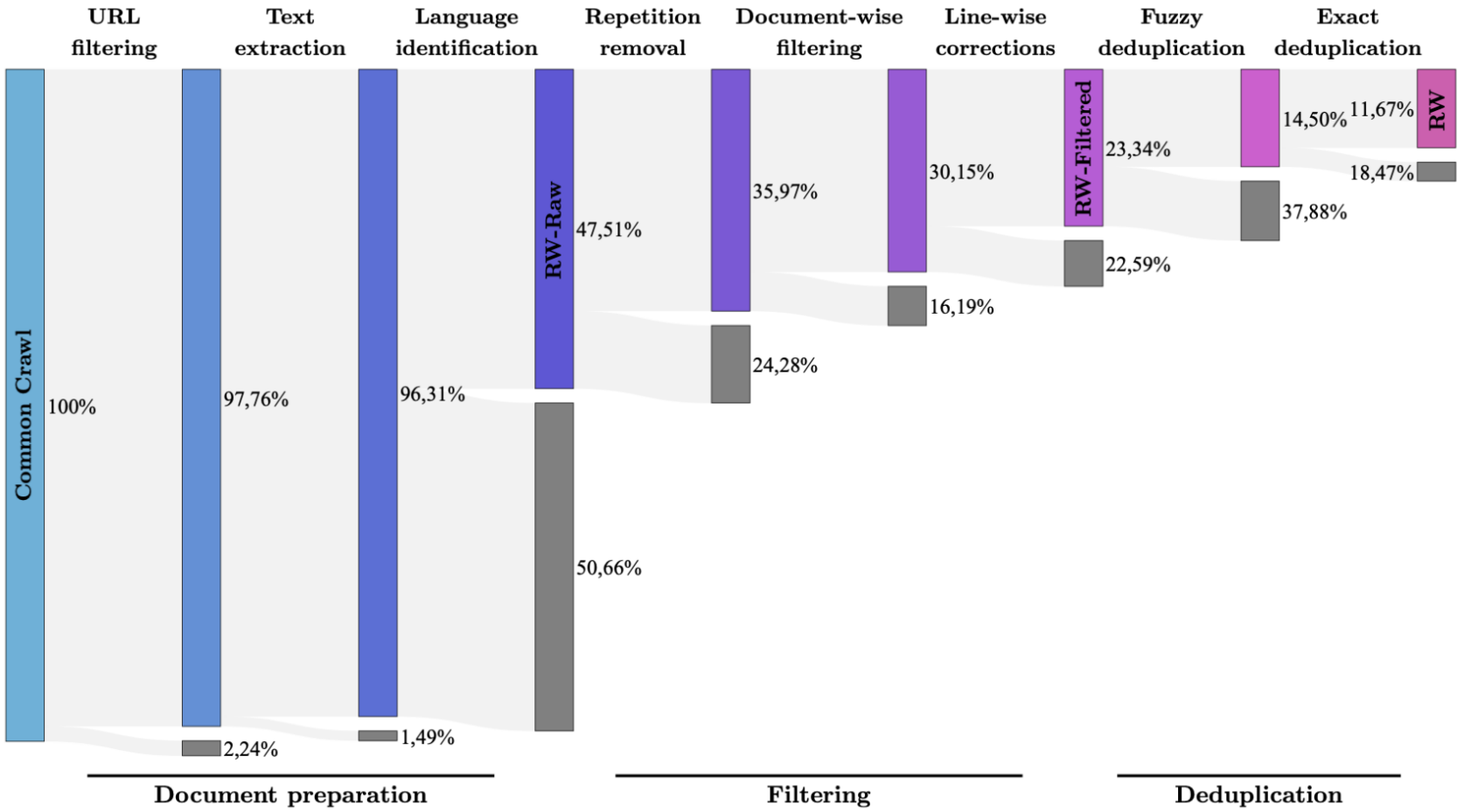
Ben Lengerich © University of Wisconsin-Madison 2026

Why does this work? Some hypotheses

- Task identification?
  - Xie et al. (2021). An explanation of in-context learning as implicit Bayesian inference
  - Raventos, et al. (2023). Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression
- Some kind of ”learning” without model updates?
  - Akyurek, et al. (2024). In-context language learning: architectures and algorithms
  - von Oswald, et al. (2023). Transformers learn in-context by gradient descent
- Both?
  - Pan, et al. (2023). What in-context learning ”learns” in-context: disentangling task recognition and task learning

Pretraining in practice

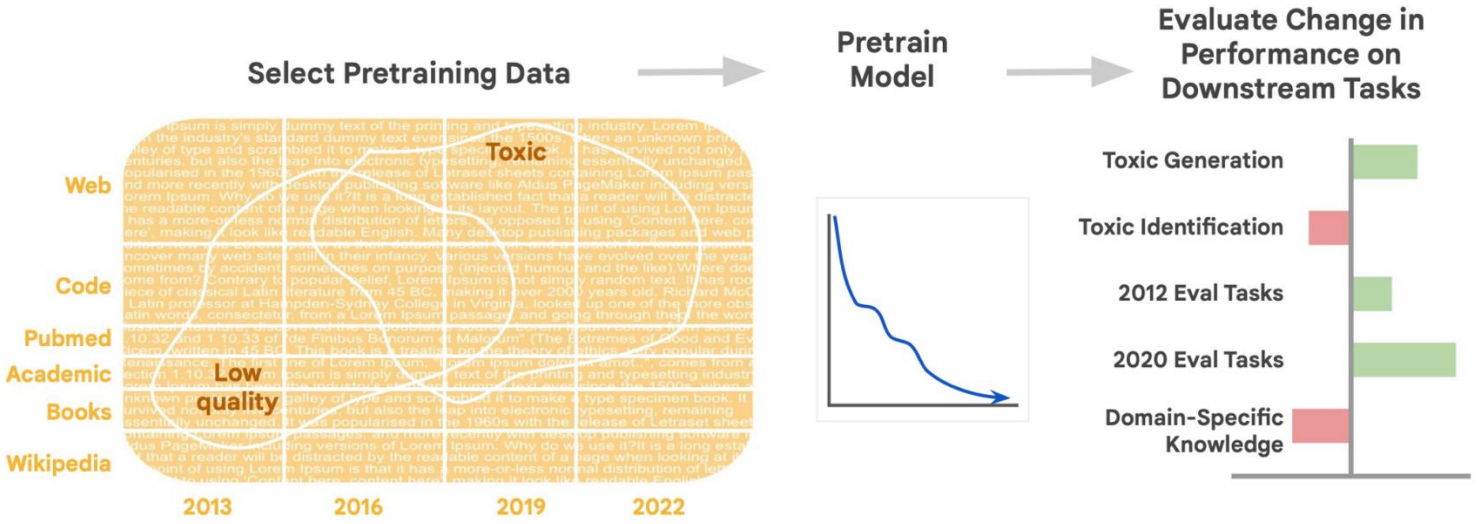
Scale is difficult…for example, data filtering



Penedo, et al. (2023) The Refined Web dataset for Falcon LLM

Scale is difficult…for example, data filtering

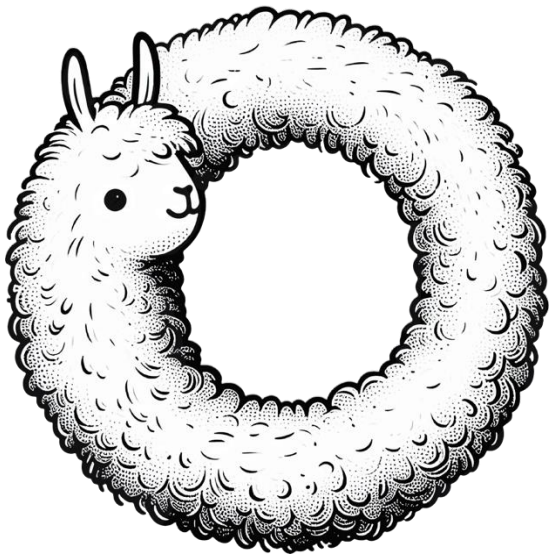
A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity [Longpre et al, NAACL 2024]



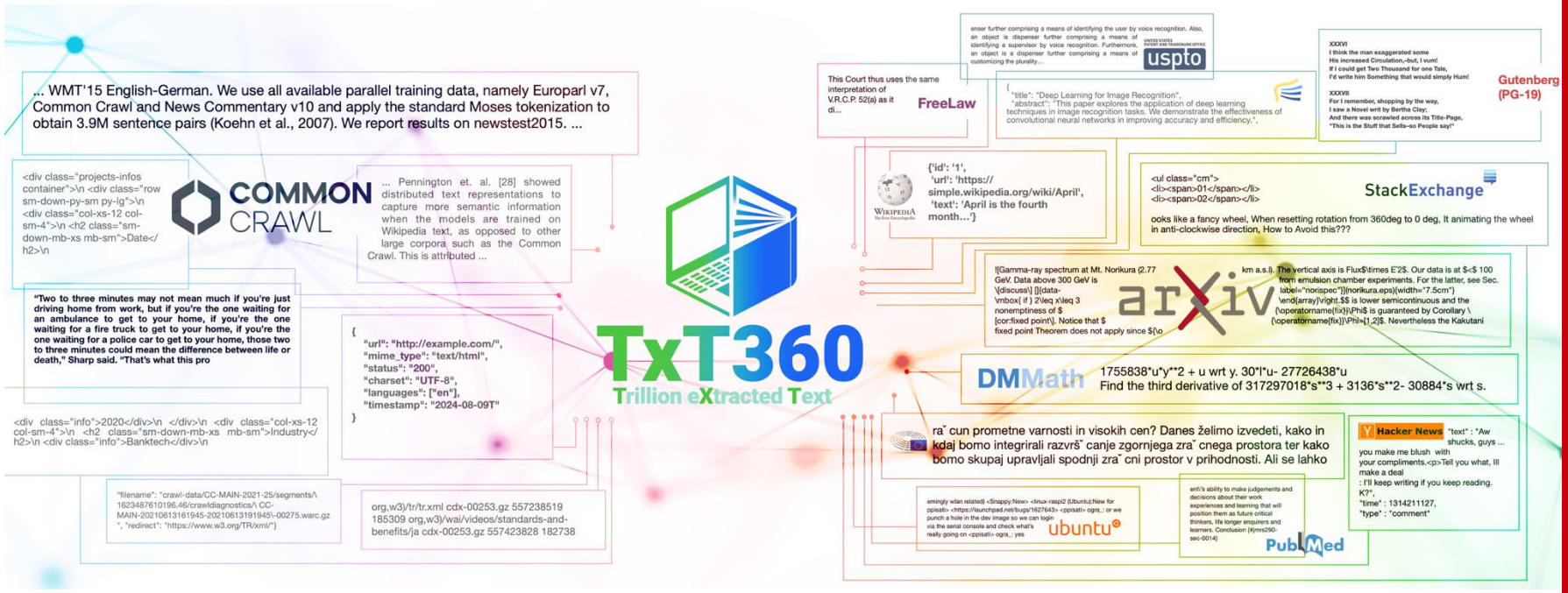
Some findings: strongly encourage to read the paper!

- "temporal shift between evaluation data and pretraining data leads to performance degradation, which is not overcome by finetuning"
- "a trade-off between performance on standard benchmarks and risk of toxic generations… there does not exist a one-size-fits-all solution to filtering."

A great learning opportunity



LLM360



[TxT360]

Memory math: why one GPU is not enough

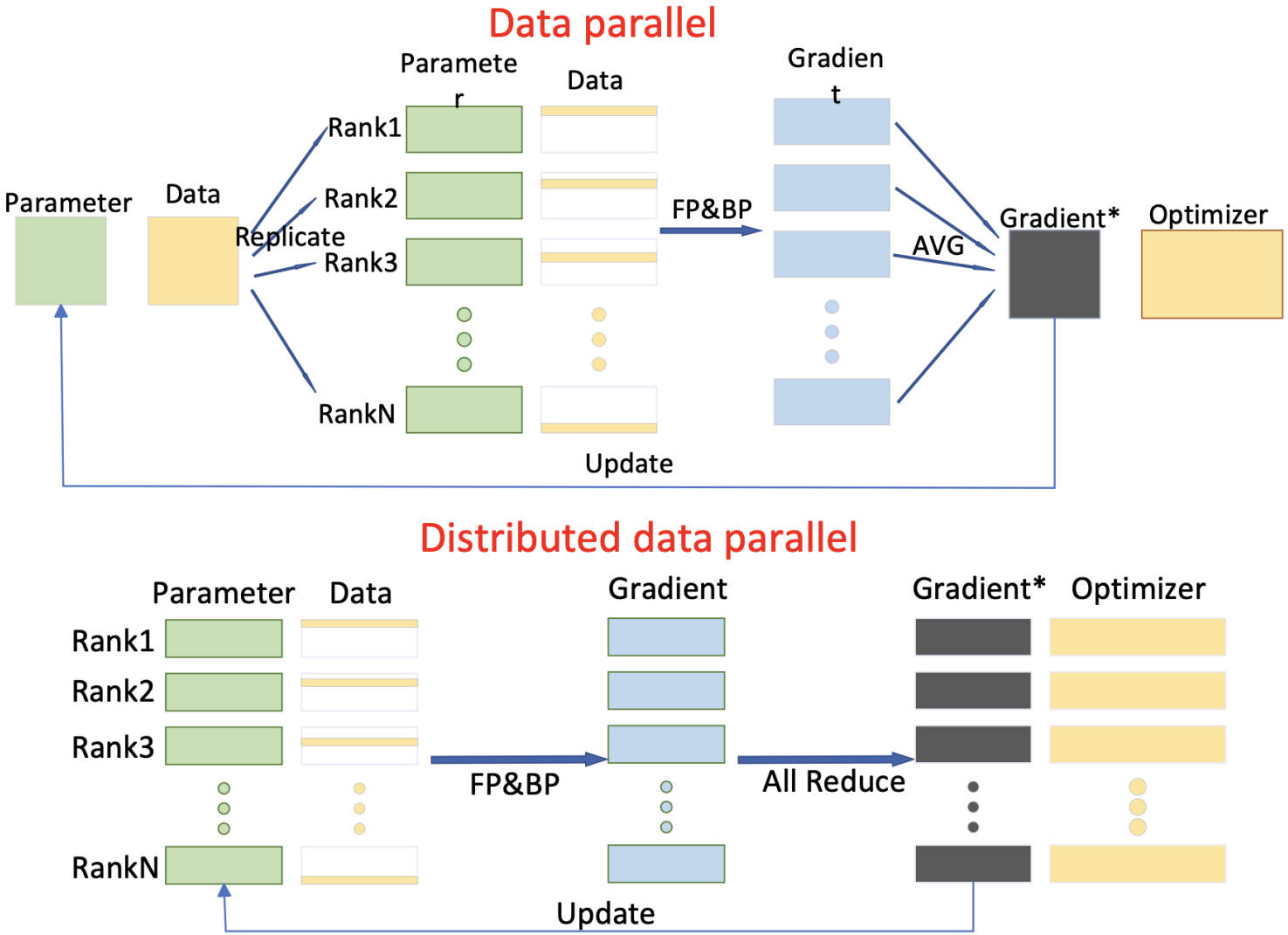
$$\text{bytes} \approx \underbrace{2}_{\text{bf16 weights}} + \underbrace{2}_{\text{grads}} + \underbrace{12}_{\text{fp32 Adam: copy, } m, v} = 16 \text{ bytes per parameter}$$

Model	Training state	80 GB GPUs needed
1B	16 GB	1
7B	112 GB	2
70B	1.1 TB	14
405B	6.5 TB	81

- …and that is **before** activations, which grow with batch size × sequence length
- So the model itself must be split across GPUs, not just the data

Ben Lengerich © University of Wisconsin-Madison 2026

Scale is difficult…for example, parallel training



[Understanding LLMs: A Comprehensive Overview from Training to Inference]

Splitting the work across GPUs

Strategy	What is split	Each GPU holds	Main cost
Data parallel	the batch	a full copy of the model	average gradients every step
ZeRO / FSDP	optimizer state, gradients, weights	1/G of the training state	gather weights layer by layer
Tensor parallel	each weight matrix	a slice of every layer	communication inside every layer
Pipeline parallel	the layers	a contiguous stack of layers	idle “bubbles” between micro-batches
Expert parallel (MoE)	the experts	a subset of experts	all-to-all routing of tokens

- Real runs combine them (“3D parallelism”): e.g., tensor within a node, pipeline across nodes, data / FSDP across everything

Next Time: Aligning LLMs

SFT, RLHF, and DPO

From a model of text to a model that follows instructions

Ben Lengerich © University of Wisconsin-Madison 2026

Questions?

