STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 18: GPT Architectures

November 10, 2026

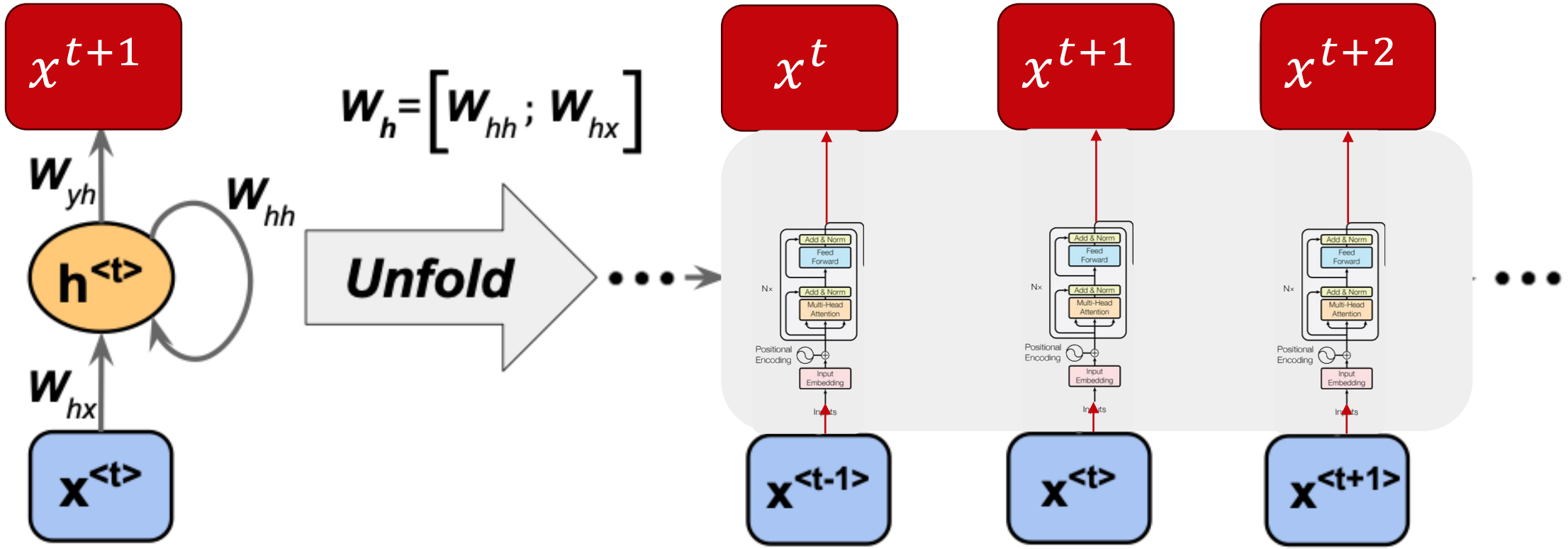
Reading: See course homepage

Logistics

- **HW4 out today**, due Fri 11/20: VAEs, diffusion, and GANs
- Project presentations: Tue 12/1 and Thu 12/3
  - **Sign up for a slot** on the sign-up sheet
  - 4-minute talks; upload slides to Canvas 24 hours before your slot

Ben Lengerich © University of Wisconsin-Madison 2026

Last time: From RNN…to GPT…by Transformers



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Last time: one decoder block

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V, \qquad M_{ij} = \begin{cases} 0 & j \le i \\ -\infty & j > i \end{cases}$$

$$\mathbf{h} = \mathbf{x} + \text{MultiHead}(\text{LN}(\mathbf{x}))$$

$$\text{out} = \mathbf{h} + \text{FFN}(\text{LN}(\mathbf{h}))$$

- Attention mixes information **across** tokens; the MLP processes each token **on its own**
- Residual connections + LayerNorm let us stack N of these blocks
- Causal mask: one forward pass trains next-token prediction at all T positions in parallel

Ben Lengerich © University of Wisconsin-Madison 2026

Today: GPT

Generative Pre-Trained Transformer

Today

- **From Transformer to GPT:** a probabilistic model + a stack of decoder blocks
- **Anatomy of a GPT:** shapes, parameters, generation, KV cache
- **Let's write a GPT:** EasyGPT, ~90 lines
- **From GPT-1 to today's GPTs:** tokenizers, scale, the modern recipe
- **Mixture of Experts:** a 1991 probabilistic idea inside 2026 LLMs

Ben Lengerich © University of Wisconsin-Madison 2026

From Transformer to GPT

From Sequence Transduction to Sequence Modeling

- **Original Transformer** (Vaswani et al., 2017):

$$P(Y \mid X) = \prod_t P(Y_t \mid Y_{<t}, X)$$

  - **Conditional** sequence model for tasks like translation (input → output)
- **Generative Pretrained Transformer (GPT)** Models:

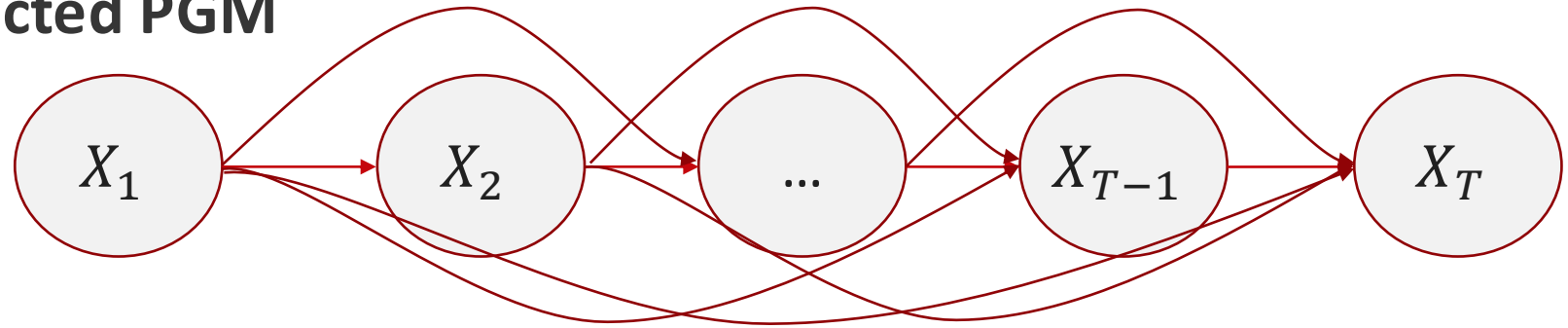
$$P(X) = \prod_t P(X_t \mid X_{<t})$$

  - **Unconditional** generative model over raw text
- Architectural consequence: **no encoder**, only a decoder with causal structure

Ben Lengerich © University of Wisconsin-Madison 2026

GPT = Probabilistic Model + Transformer Decoder

- **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

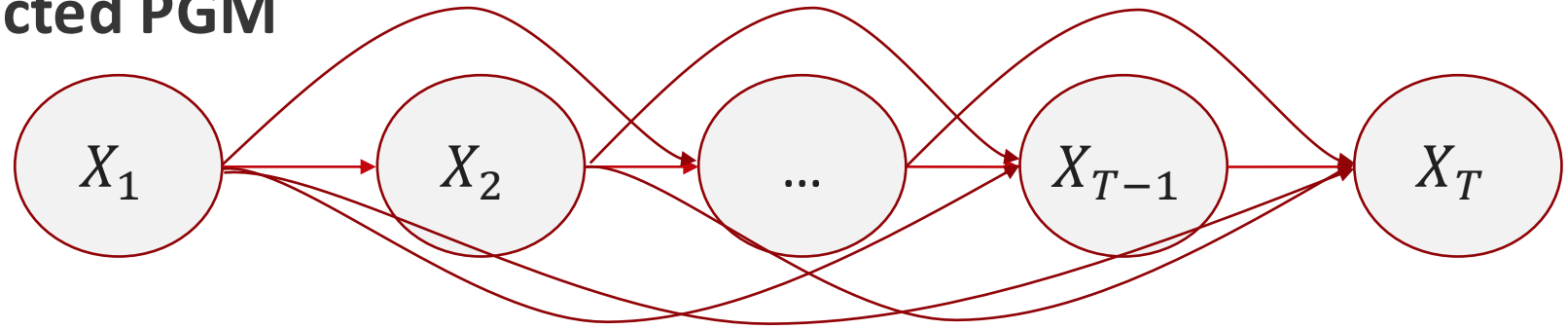
$$\max_\theta \sum_i \sum_t \log P_\theta\left(X_{i,t} \mid X_{i,<t} \right)$$

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

Ben Lengerich © University of Wisconsin-Madison 2026

GPT = Probabilistic Model + Transformer Decoder

- **Directed PGM**



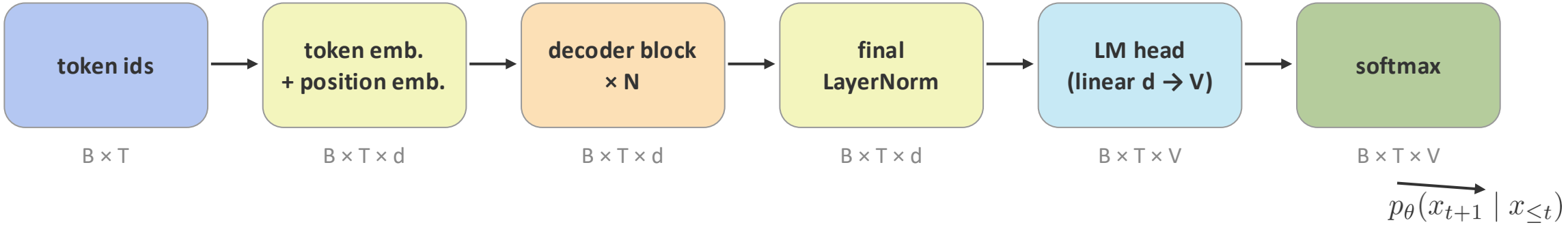
$$P_\theta(X) = \prod_i \prod_t P_\theta(X_{i,t} \mid X_{i,<t})$$

- **Model structure:**
  - Input: token embeddings + positional encodings
  - Masked multi-head attention: Enforces “causality”
  - Decoder stack: Learns $P(X_t \mid X_{<t})$
  - Output: softmax over vocabulary

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

Ben Lengerich © University of Wisconsin-Madison 2026

A GPT, end to end

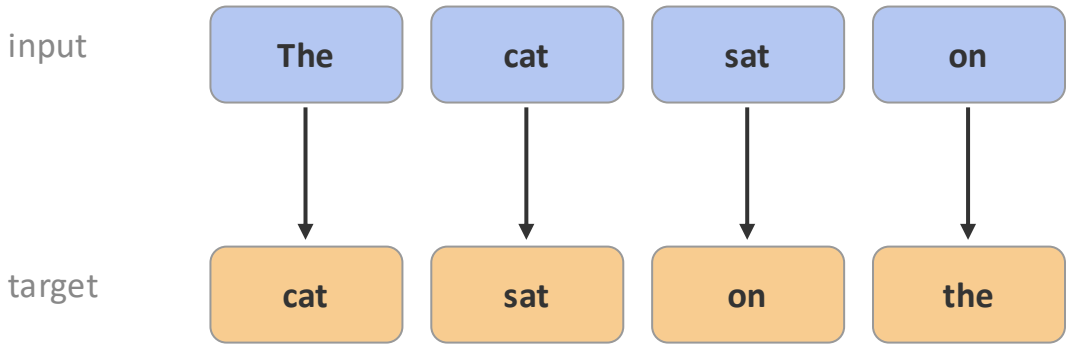


- **One** forward pass gives a next-token distribution at **every** position (causal mask)
- **Training:** score each position against the true next token
- **Generation:** use only the **last** position

The output layer and the loss

$$\mathbf{z}_t = \mathbf{h}_t^{(N)} W_E^\top \in \mathbb{R}^V, \qquad p_\theta(x_{t+1} = v \mid x_{\le t}) = \mathrm{softmax}(\mathbf{z}_t)_v$$

$$\mathcal{L}(\theta) = -\frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \log p_\theta(x_{b,t+1} \mid x_{b,\le t})$$



- Targets = inputs shifted by one token
- **Weight tying:** one V × d matrix both embeds tokens and scores them
- Cross-entropy = average negative log-likelihood = the MLE objective
- Sanity check: initial loss ≈ ln V (4.17 for 65 characters)

Ben Lengerich © University of Wisconsin-Madison 2026

Where do the parameters live?

$$N \approx \underbrace{12\,L\,d^2}_{\text{blocks: } 4d^2 \text{ attention} + 8d^2 \text{ MLP}} + \underbrace{V d}_{\text{token emb.}} + \underbrace{T_{\max} d}_{\text{pos. emb.}}$$

Model	L	d	12·L·d²	V·d	Total	Usually quoted
GPT-0 (ours)	4	64	0.20M	0.004M	0.21M	208,961
GPT-1	12	768	84.9M	31.1M	116M	117M
GPT-2 small	12	768	84.9M	38.6M	124M	124M
GPT-2 XL	48	1600	1.47B	0.08B	1.56B	1.5B
GPT-3	96	12,288	173.9B	0.6B	174.6B	175B

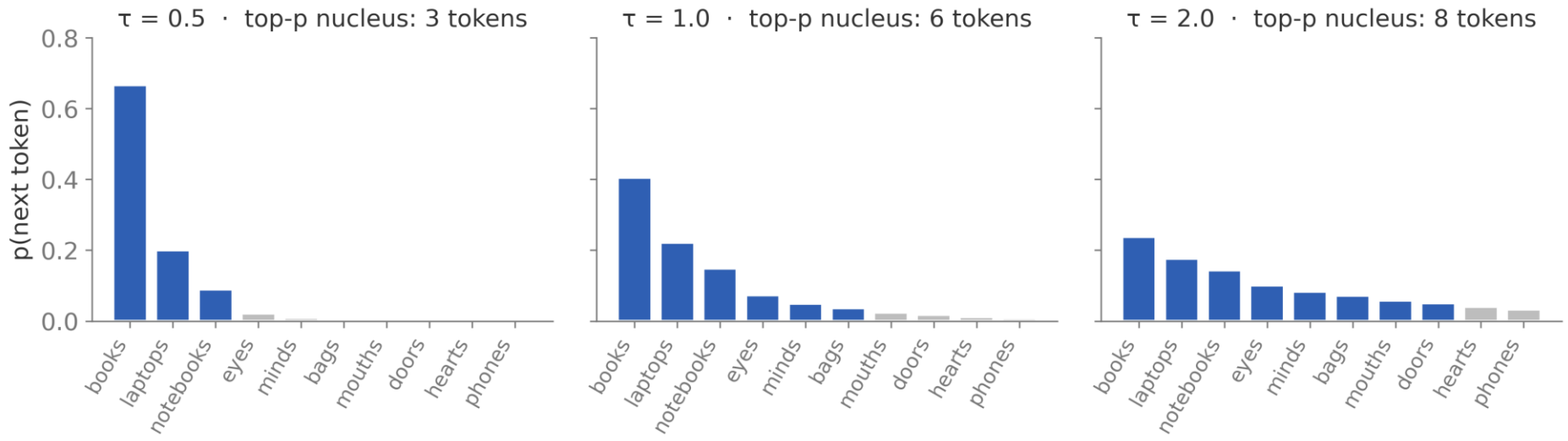
- Small models: embeddings are a big share
- Large models: >99% of parameters are in the blocks, and **2/3 of those are in the MLPs**
- Compute per token ≈ 2N FLOPs forward, ≈ 6N to train (Thursday)

Generating text, one token at a time

```
for _ in range(max_new_tokens):
    logits = model(idx[:, -T:])[:, -1]
    probs  = softmax(logits / τ)
    nxt    = sample(probs)   # or argmax
    idx    = cat(idx, nxt)
```

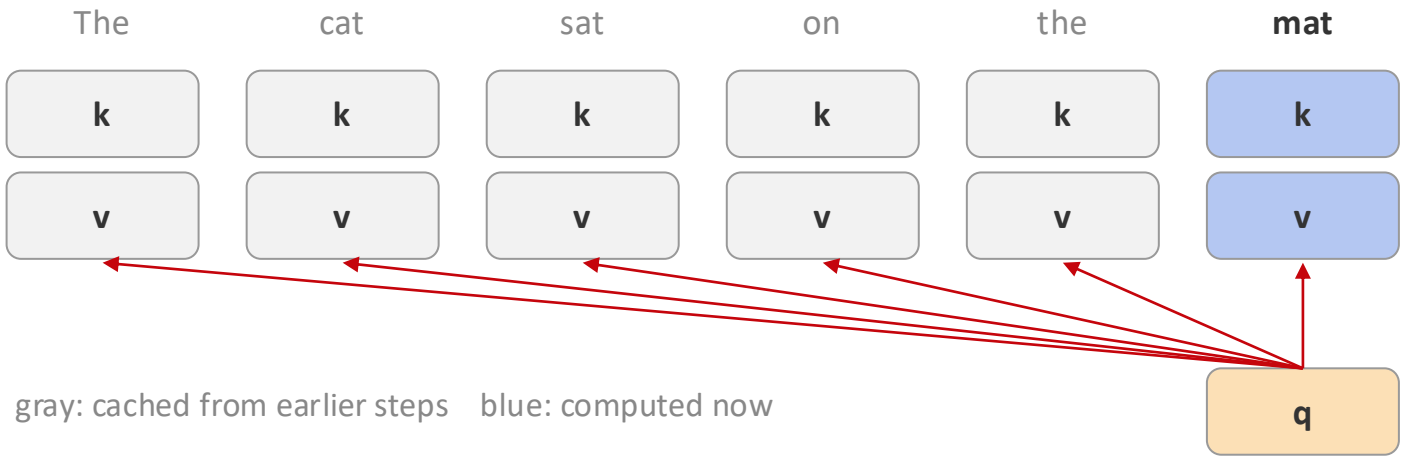
$$p_\tau(v) = \frac{\exp(z_v/\tau)}{\sum_u \exp(z_u/\tau)}$$

- **Greedy:** argmax every step ($\tau \to 0$)
- **Temperature τ:** < 1 sharper, > 1 flatter
- **Top-k / top-p:** sample from the k likeliest tokens / the smallest set with mass ≥ p
- **Why one at a time?** Each step's input is the token just sampled, which doesn't exist until the previous step finishes



Ben Lengerich © University of Wisconsin-Madison 2026

KV cache: don’t recompute the past



- Without a cache: step t re-runs all t tokens through every layer
- With a cache: step t runs **one** token; its query reads the stored keys and values

$$\text{KV cache} = 2 \times L \times d \times T \times \text{bytes per value}$$

- GPT-3: 2 × 96 × 12,288 × 2 bytes ≈ **4.7 MB per token** → ≈ 9.7 GB for one 2,048-token context
- Generation is limited by memory, not arithmetic → motivates GQA and MLA (later today)

Ben Lengerich © University of Wisconsin-Madison 2026

Let’s write a GPT

Let’s write a GPT

- EasyGPT repo
  - Adapted from Andrej Karpathy’s nanoGPT
  - gpt0.ipynb: character-level GPT on Tiny Shakespeare (≈ 1 MB of text)
  - gpt_zero.py: the whole model in ~90 lines: embeddings → blocks → LayerNorm → LM head
  - transformer.py: causal self-attention + MLP block (the one from last lecture)
  - Config: 4 layers, 4 heads, d = 64, context 8, vocab 65 characters → 208,961 parameters

Ben Lengerich © University of Wisconsin-Madison 2026

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

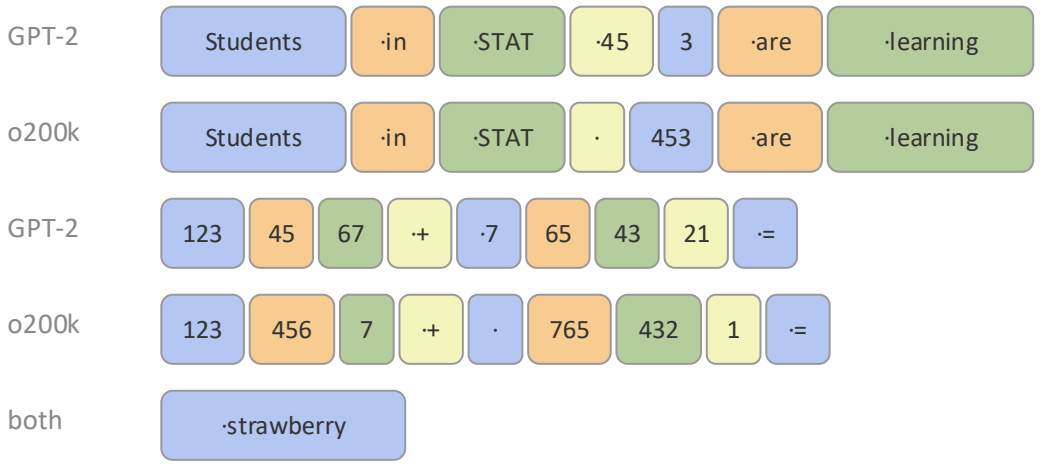
Tokenization: byte-pair encoding (BPE)

- 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 (Thursday)

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

Same skeleton as GPT-2; the rows show what changed.

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

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

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

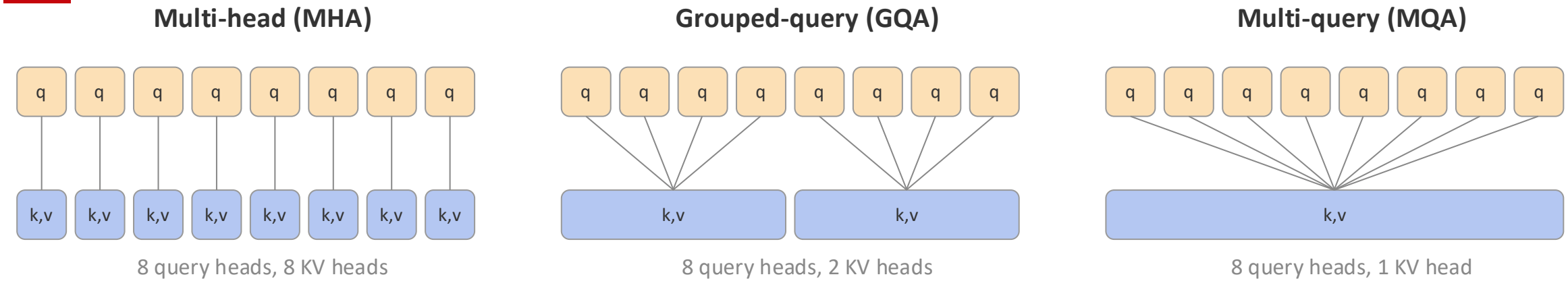
Unchanged since 2018:

decoder-only

causal self-attention + MLP + residual stream

next-token cross-entropy pretraining

Grouped-query attention: shrinking the KV cache



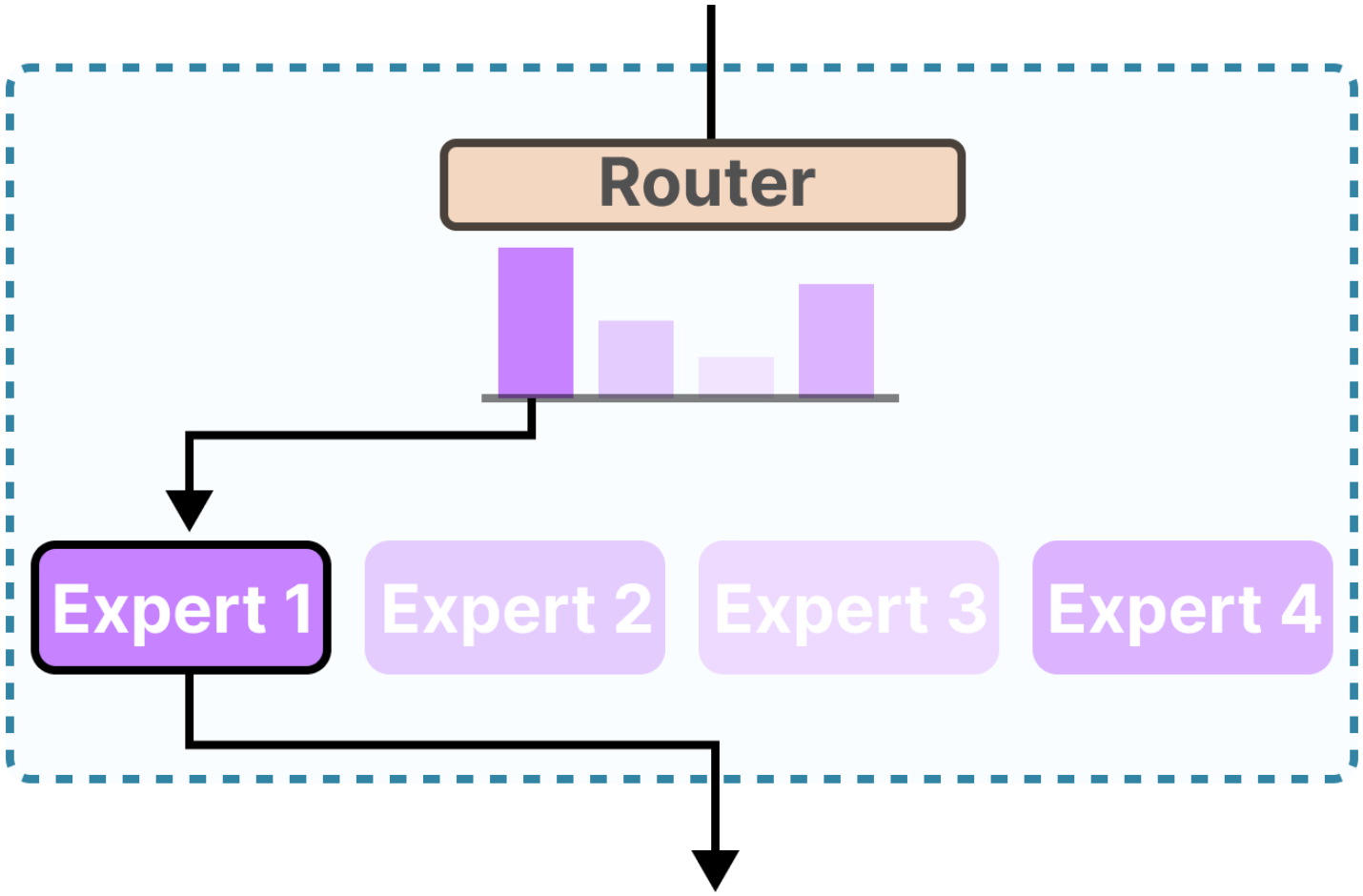
$$\text{KV cache} = 2 \times L \times n_{\text{kv}} \times d_{\text{head}} \times T \times \text{bytes}$$

- Llama 3 70B at 128K tokens (bf16): **344 GB** with 64 KV heads (MHA) vs **43 GB** with 8 (GQA), for one sequence
- Small quality cost; much faster generation (each step reads far less memory)
- DeepSeek's **MLA**: cache one small latent vector per token; rebuild K, V from it

Ben Lengerich © University of Wisconsin-Madison 2026

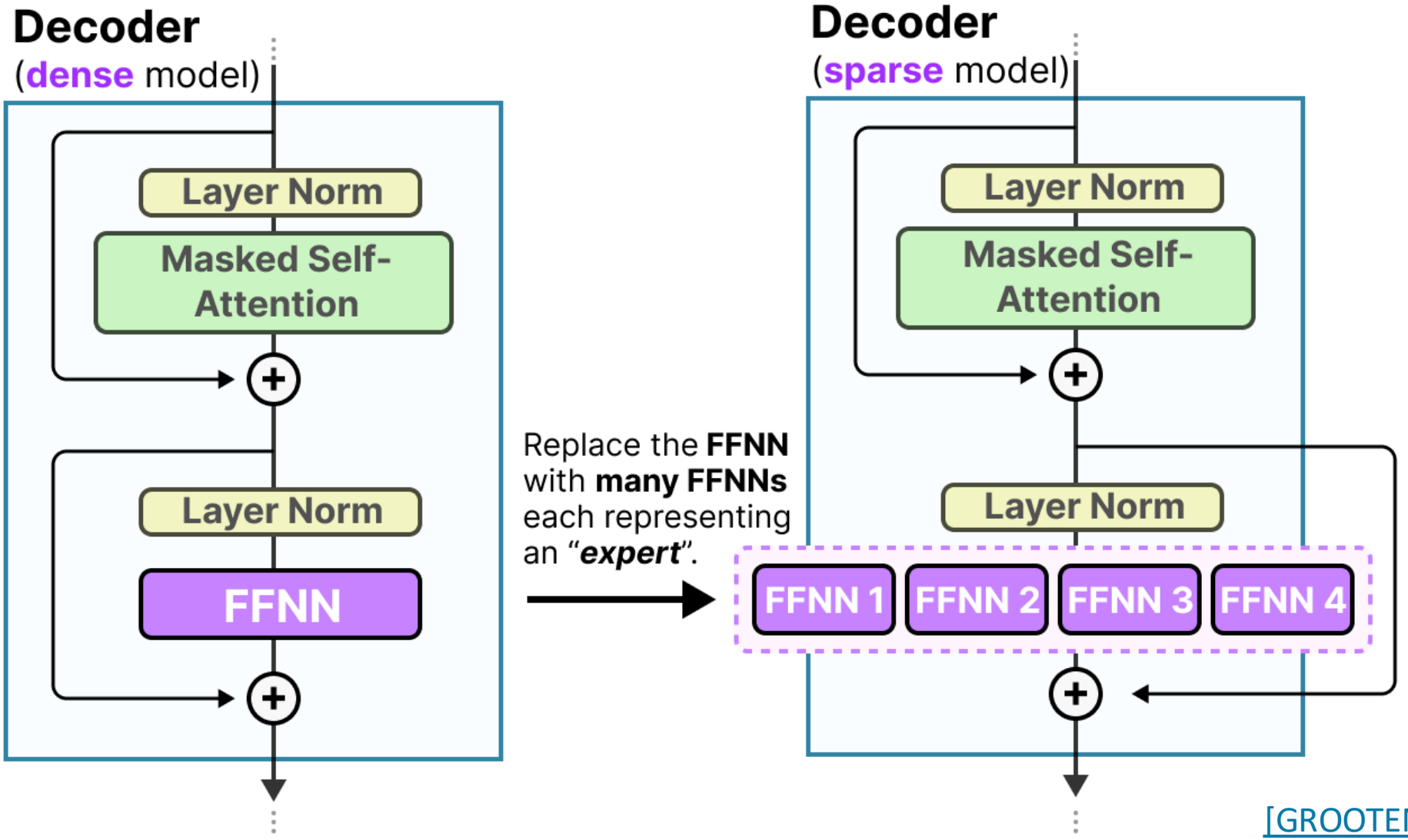
Mixture of Experts

Mixture of Experts



[GROOTENDORST]

Mixture of Experts inside Transformer Decoder



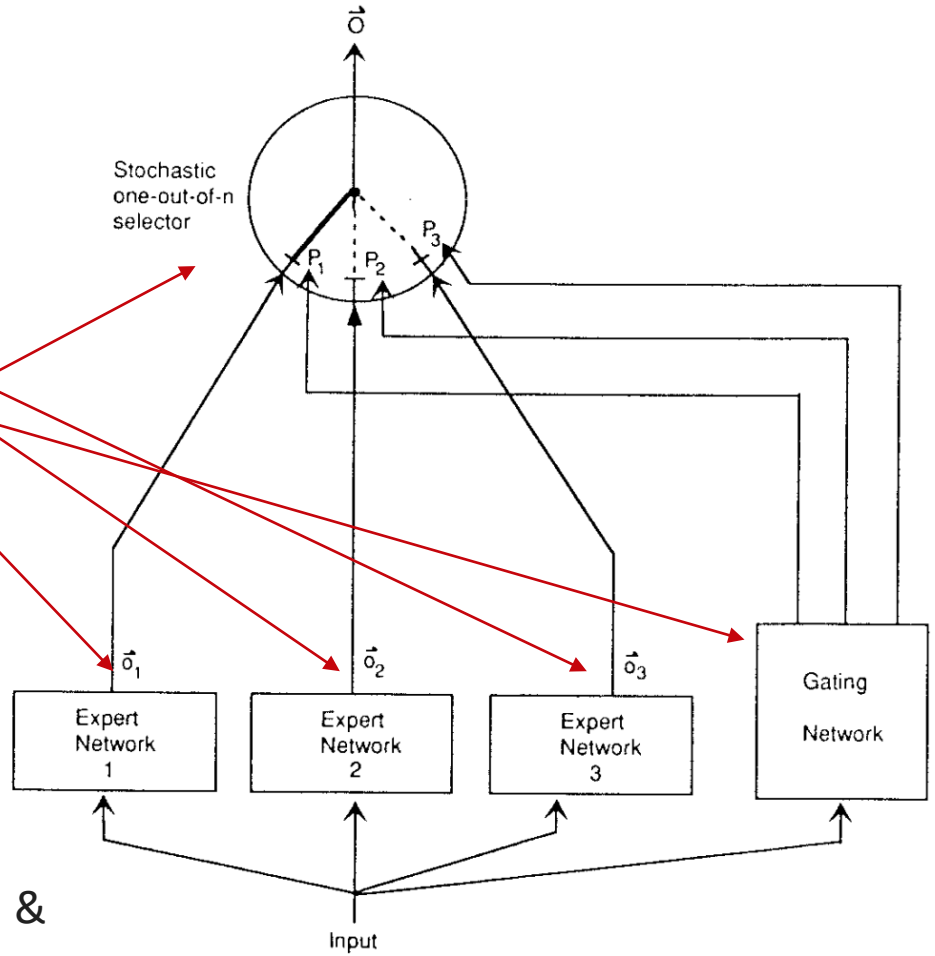
[GROOTENDORST]

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"

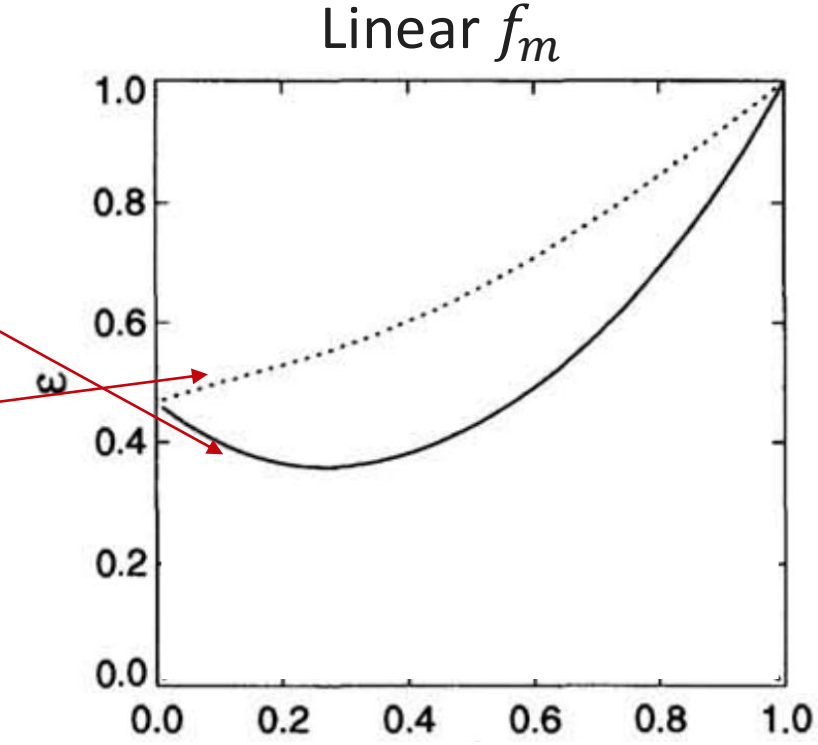
Will these be minimized for the same ensemble?

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]

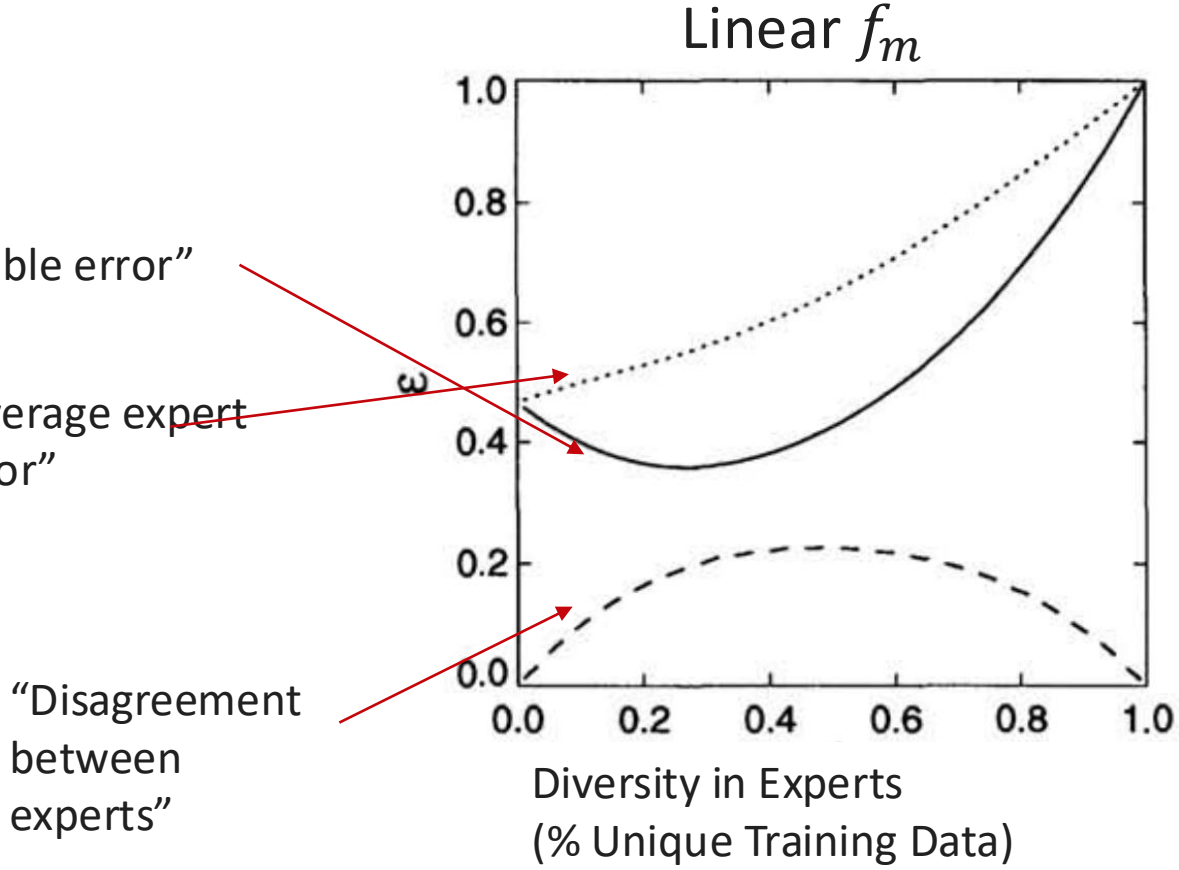
Bonus

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]

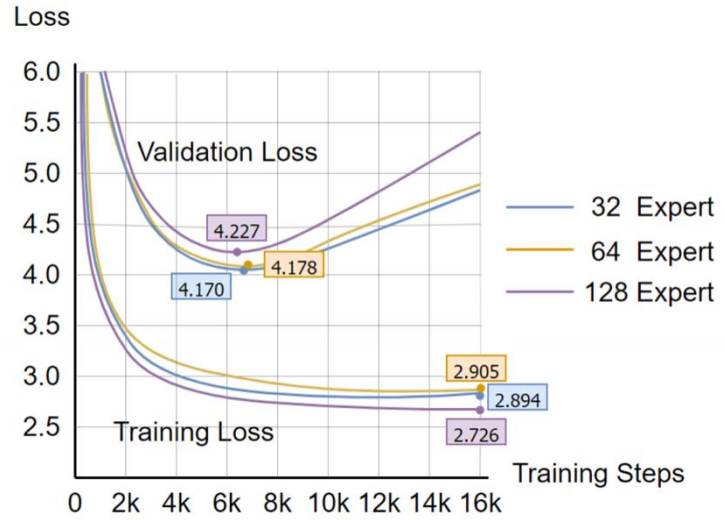
Ben Lengerich © University of Wisconsin-Madison 2026

Mixture of Experts: In LLMs

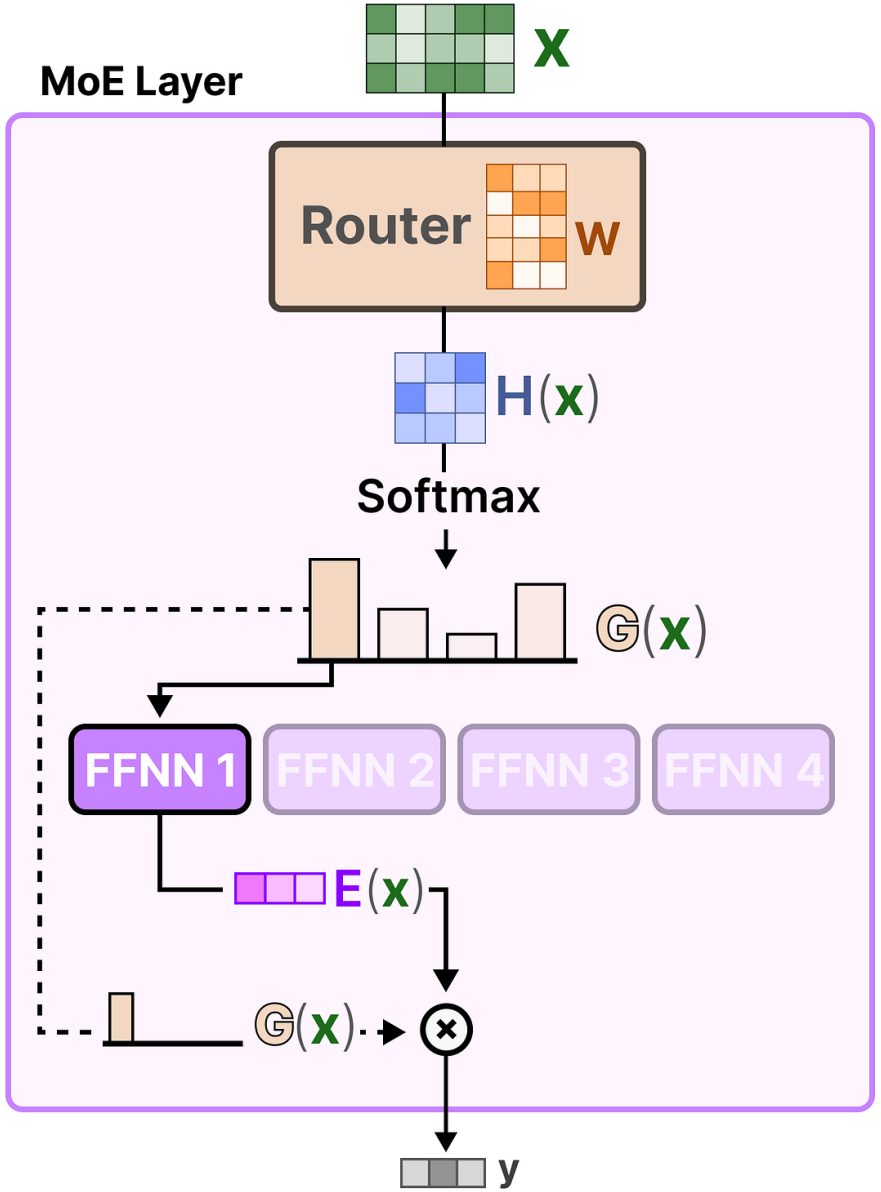
- Implications for serving?

- 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 (Thursday)
  - Then post-training: SFT + RL from human feedback (next week)

Next Time: Unsupervised Training of LLMs

Data, scale, and what maximum likelihood buys you

Ben Lengerich © University of Wisconsin-Madison 2026

Questions?

