IAIRO
Indian AI Research Organisation

PRAMANA: SLM++ BOOTCAMP · COHORT 1 · SESSION 3

Modern SLM Upgrades & Mixture of Experts

Five changes turn the GPT-2 block into a Llama 3 block. Each one answers a specific failure.

Mohit Joshi

Pretraining Researcher – IAIRO

IN PARTNERSHIP WITH

Department of Science & Technology (Government of Gujarat)

INDIAai

Anusandhan National Research Foundation

सत्यमेव जयते

इलेक्ट्रॉनिकी एवं सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF ELECTRONICS AND INFORMATION TECHNOLOGY

AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

Ahmedabad University

NVIDIA

ABOUT THE SPEAKER

Mohit Joshi

Researcher · IAIRO
mhtjsh.github.io

Pre-training Research and Convex Optimization Research
Deep Knowledge Infusion Architectures

120 MINUTES

Session agenda

Five upgrades, each in the same three beats: what existed, what broke, what fixed it.

00:00	Framing: the block you already built	10 min
00:10	Normalization - placement, then RMSNorm	22 min
00:32	Position - absolute embeddings to RoPE	24 min
00:56	Attention cost - MHA to GQA	22 min
01:18	Feed-forward - GELU MLP to SwiGLU	14 min
01:32	Mixture of Experts - routing and its price	20 min
01:52	**Guided lab, synthesis and questions**	8 min

Five Upgrades, Three Issues.

Same residual block. Same tensor shapes in and out. Only the five majorly marked components change.

BY THE END OF THIS SESSION

Learning objectives

Each objective is testable in the lab notebook.

01

Diagnose: Prior Work

Given a decoder block, name which failure each component was introduced to fix.

02

Derive: What Broke

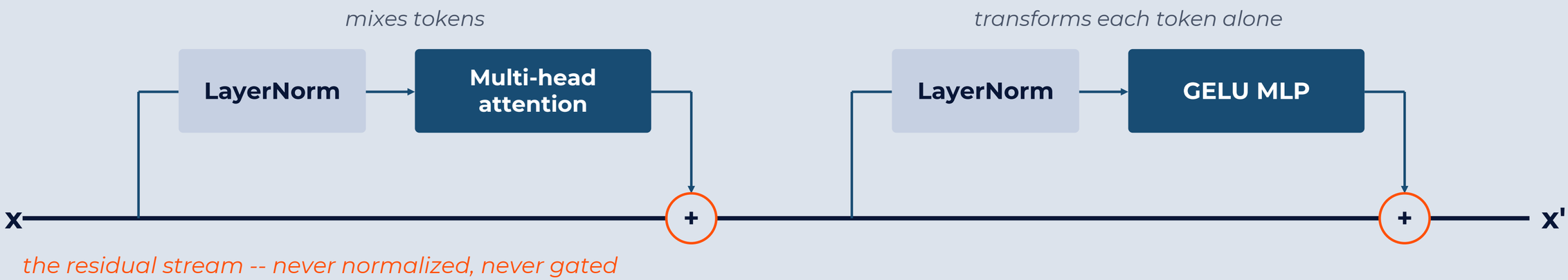
Write the equation for pre-norm, RMSNorm, RoPE, GQA cache size, SwiGLU and top-k routing.

03

Refactor: Fix

Rebuild your Day 2 GPT-2 block to Llama 3 spec and verify it with shape, causality and router tests.

GPT 2 Blocks from Day 2

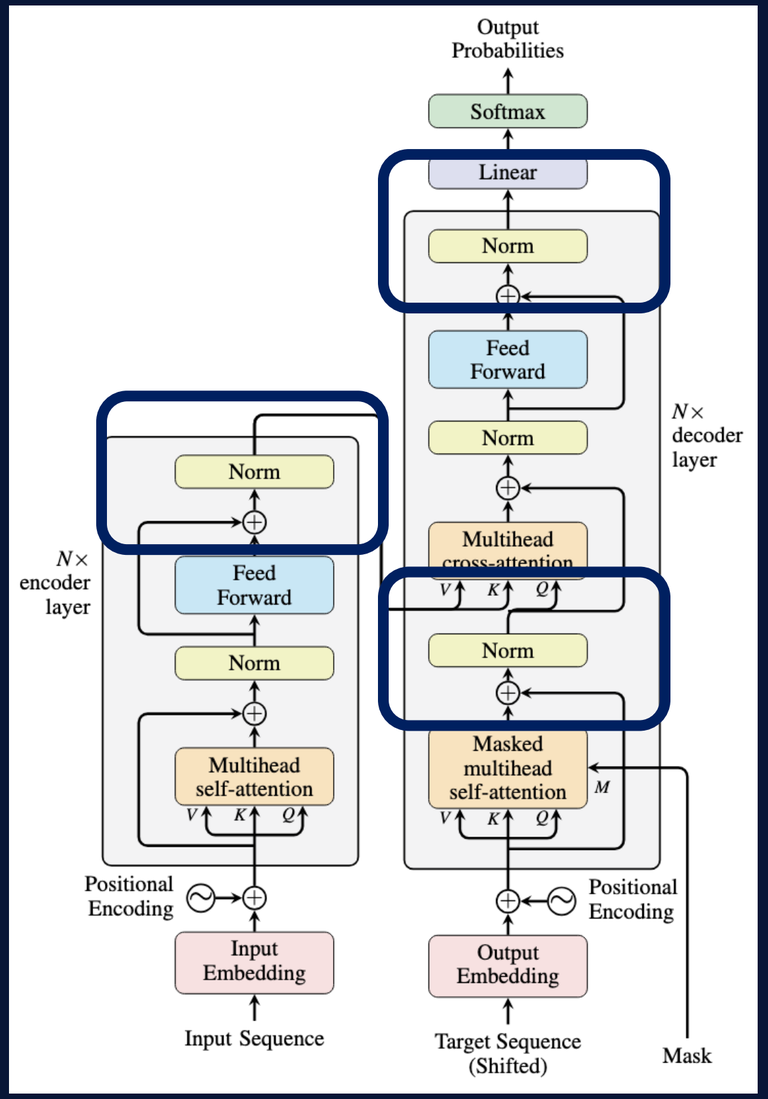


POSITION	NORMALIZATION	ATTENTION	FEED-FORWARD
learned table, added once at layer 0	LayerNorm, on the branch input	12 query heads, 12 key/value heads	d → 4d → d, GELU

Radford et al. (2019), Language Models are Unsupervised Multitask Learners. OpenAI technical report.

01
Pre Norm: Make depth survivable

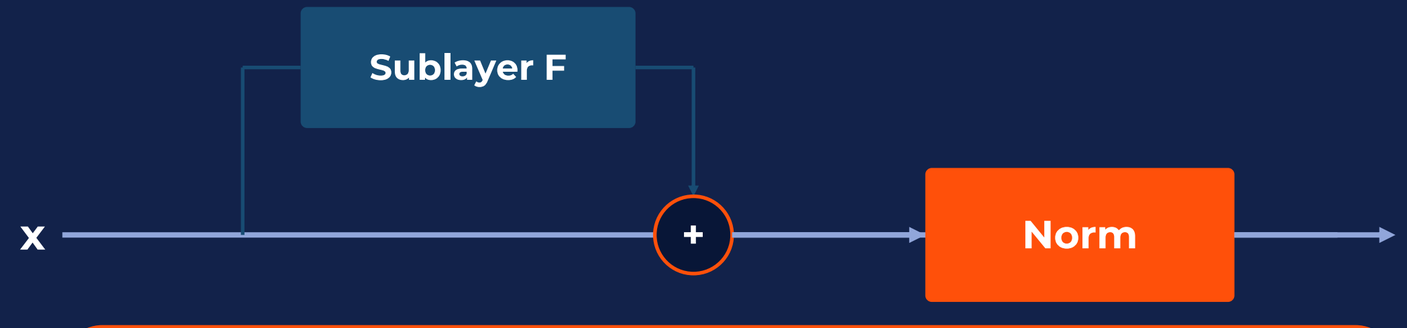
Where normalization sits, and how much of it you need.



2017: normalize after the addition

BEFORE | WHAT BREAKS | THE FIX

The original transformer wiring



$$x_{l+1} = \mathrm{Norm}\big(x_l + F(x_l)\big)$$

Every sublayer output is renormalized before it continues.

Needed learning-rate warmup to train at all

the norm sits ON the highway, not beside it

Depth ceiling

Original models stopped near 6–12 layers per stack

Warmup required

Helps in normalizing the depths before each layers

Tuning sensitivity

Results move sharply with warmup length

Vaswani et al. (2017), Attention Is All You Need. NeurIPS 2017.

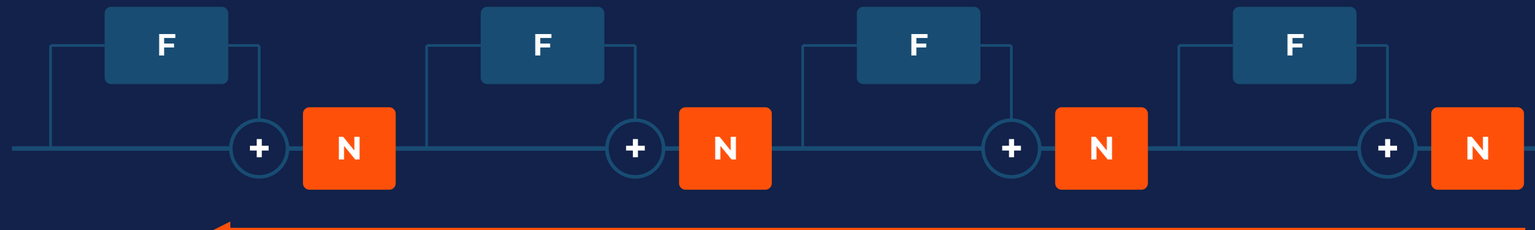
Follow the gradient backwards

BEFORE | WHAT BREAKS | THE FIX

Same blocks, different backward path (follow gradient Backwards

POST-NORM

norm inside the residual path



PRE-NORM

norm on the branch only

In post-norm the norm's Jacobian multiplies the gradient once per layer, so the product compounds with depth. In pre-norm the highway contributes an exact identity term.

Warmup becomes optional

Xiong et al. (2020), On Layer Normalization in the Transformer Architecture. ICML 2020.

The identity term is structural

Moving one box changes the Jacobian from a product into a sum.

POST-NORM

$$\frac{\partial x_{l+1}}{\partial x_l} = J_{\text{Norm}}\left(I + \frac{\partial F}{\partial x_l}\right)$$

PRE-NORM

$$\frac{\partial x_{l+1}}{\partial x_l} = I + \frac{\partial F}{\partial x_l} J_{\text{Norm}}$$

Composed over L layers, the loss gradient is a product of these:

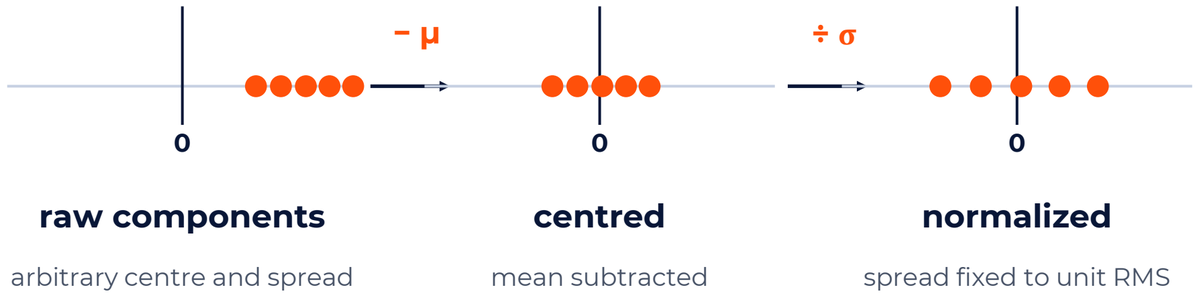
$$\frac{\partial \mathcal{L}}{\partial x_0} = \frac{\partial \mathcal{L}}{\partial x_L} \prod_{l=0}^{L-1} \frac{\partial x_{l+1}}{\partial x_l}$$

- Post-norm multiplies a normalization Jacobian into every factor of that product.
- Pre-norm guarantees each factor contains I(Identity Matrix), so the product never collapses to zero.
- The branch can still learn anything - only the collapse is avoided due to Jacobian.

Xiong et al. (2020), ICML 2020 · Radford et al. (2019), GPT-2 technical report - pre-norm adopted before it was explained.

What exactly is the norm computing?

LayerNorm does two separate things to every token vector.



$$\mathrm{LN}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} \odot \gamma + \beta$$

$$\mu = \frac{1}{n}\sum_i x_i \qquad \sigma^2 = \frac{1}{n}\sum_i (x_i - \mu)^2$$

Two statistics per token, per layer, every step

Re-centring gives invariance to shifts. Re-scaling gives invariance to gain. The 2016 paper argued both mattered - but never separated them experimentally.

Ba, Kiros & Hinton (2016), Layer Normalization. arXiv:1607.06450 (NIPS 2016 Deep Learning Symposium).

Ablation Shows:

RMS (rescaling) was load-bearing

Remove the mean subtraction and quality holds. Remove the rescaling and it does not.

LAYERNORM

$$\mathrm{LN}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} \odot \gamma + \beta$$

2 statistics · 2 learned vectors (γ, β)

RMSNORM

$$\mathrm{RMSNorm}(x) = \frac{x}{\mathrm{RMS}(x)} \odot \gamma$$

1 statistic · 1 learned vector (γ)

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{n}\sum_i x_i^2}$$

$$\mathrm{RMSNorm}(\alpha x) = \mathrm{RMSNorm}(x) \quad \forall\, \alpha > 0$$

The invariance the transformer relies on is scale invariance. RMSNorm keeps exactly that and drops the rest.

Zhang & Sennrich (2019), Root Mean Square Layer Normalization. NeurIPS 2019.

Normalization (Norm) story: up-next

BEFORE | WHAT BREAKS | THE FIX

Placement is a design space, not a settled answer

PRE-NORM

Llama 3 · Qwen 3

Norm on the branch input only.
Trainable at depth, warmup optional.

SANDWICH

Gemma 3

Norm before and after the sublayer.
Recovers post-norm quality while keeping the identity path.

QK-NORM

Qwen 3 · Gemma 3

Normalize queries and keys inside attention to stop logits growing during long runs.

$$x_{l+1} = x_l + \mathrm{Norm}_{\mathrm{post}}\Big(F\big(\mathrm{Norm}_{\mathrm{pre}}(x_l)\big)\Big)$$

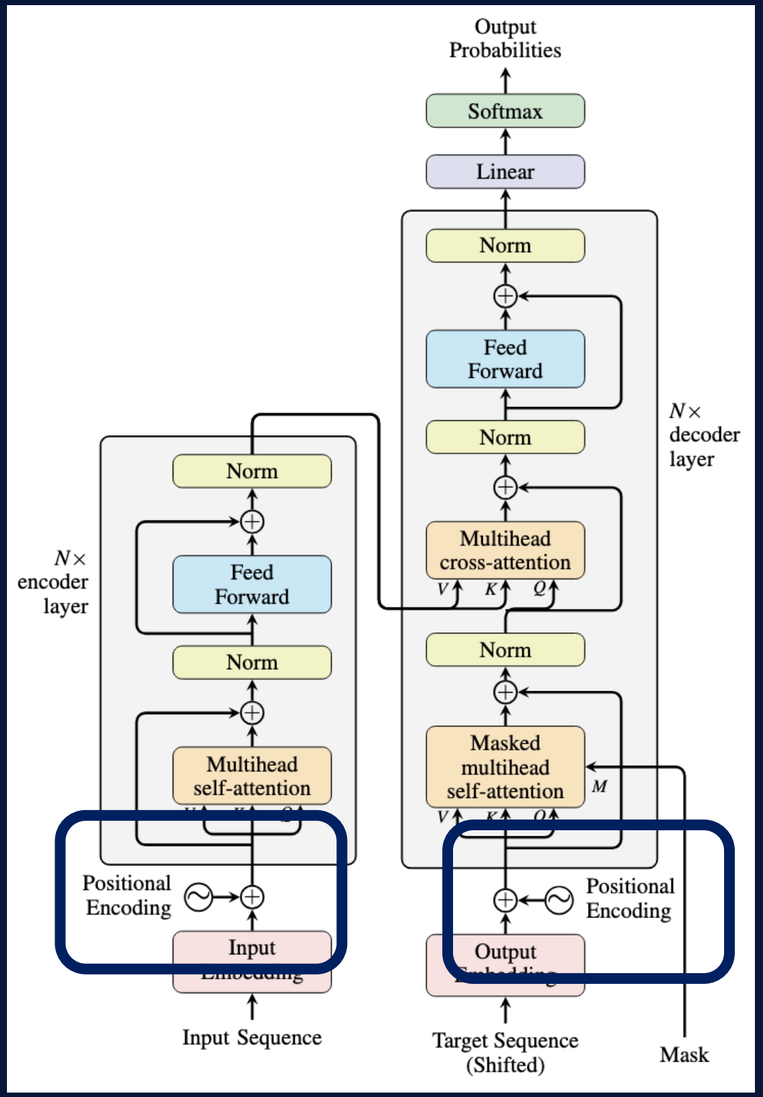
The identity path is preserved in all three. That is the invariant while tuning everything around it.

Gemma Team (2025), Gemma 3 Technical Report · Yang et al. (2025), Qwen3 Technical Report · Dehghani et al. (2023), ICML — QK-norm.

02

Put position inside attention

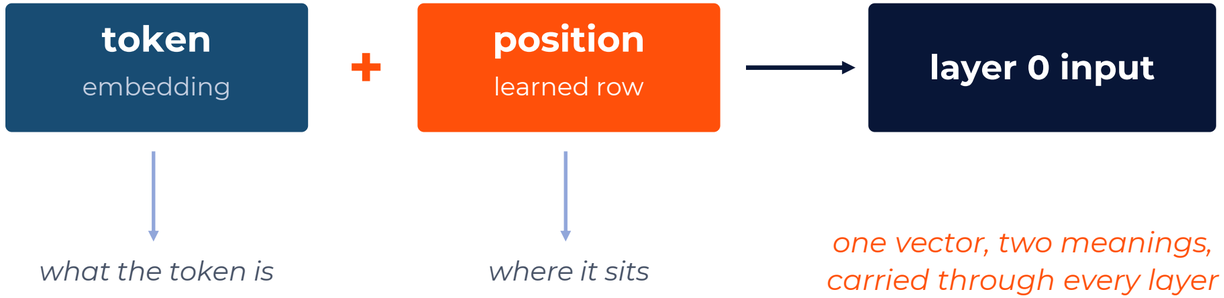
Position stopped being something you add and became something you rotate.



Position arrives once, at the very bottom

BEFORE | WHAT BREAKS | THE FIX

How GPT-2 and the original transformer both do it



$$h_i = x_i + p_i$$

Sinusoidal

Fixed cosines at geometric frequencies

Vaswani et al., 2017

Learned

A trainable row per position, capped at context length

GPT-2, 2019 — what you built

Both encode absolute position. Neither tells attention how far apart two tokens are — that has to be inferred.

Vaswani et al. (2017), NeurIPS · Radford et al. (2019), GPT-2 technical report.

Let’s investigate the attention Logit!

Three problems fall out of one line of algebra

$$\langle q_m, k_n \rangle = (x_m + p_m)^\top W_q^\top W_k (x_n + p_n)$$

$$= \underbrace{x_m^\top W x_n}_{\text{content}} + \underbrace{x_m^\top W p_n + p_m^\top W x_n}_{\text{cross terms}} + \underbrace{p_m^\top W p_n}_{\text{position}}$$

01

Entangled

Content and position multiply together in the cross terms. The model must learn to keep them apart.

02

Absolute, not relative

The logit depends on m and n separately, when what matters linguistically is n − m.

03

No extrapolation

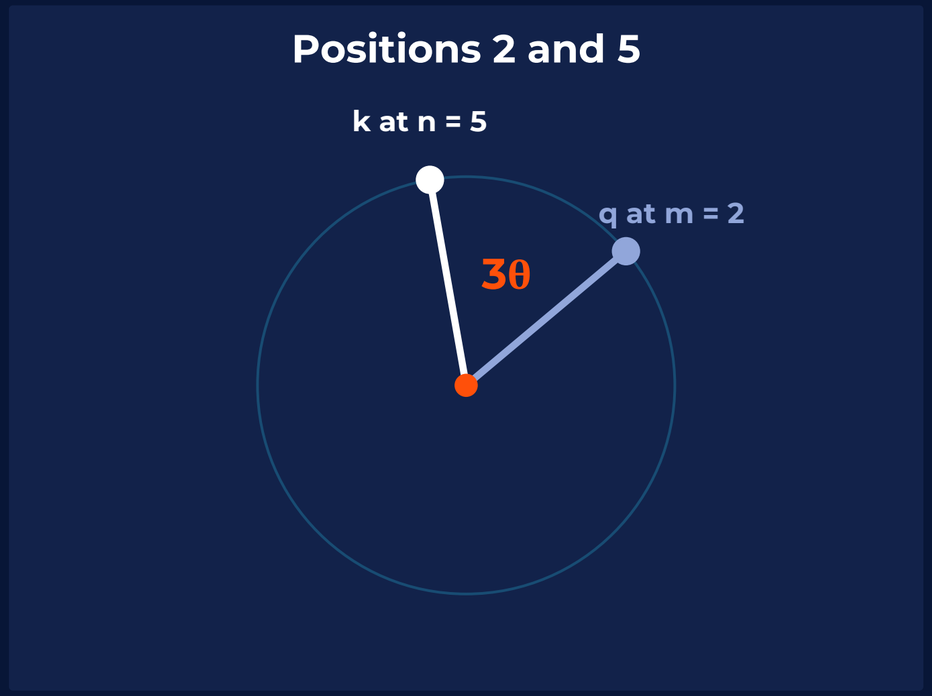
Row 4096 of a 4096-row table was never trained. Past the context length, position information is noise.

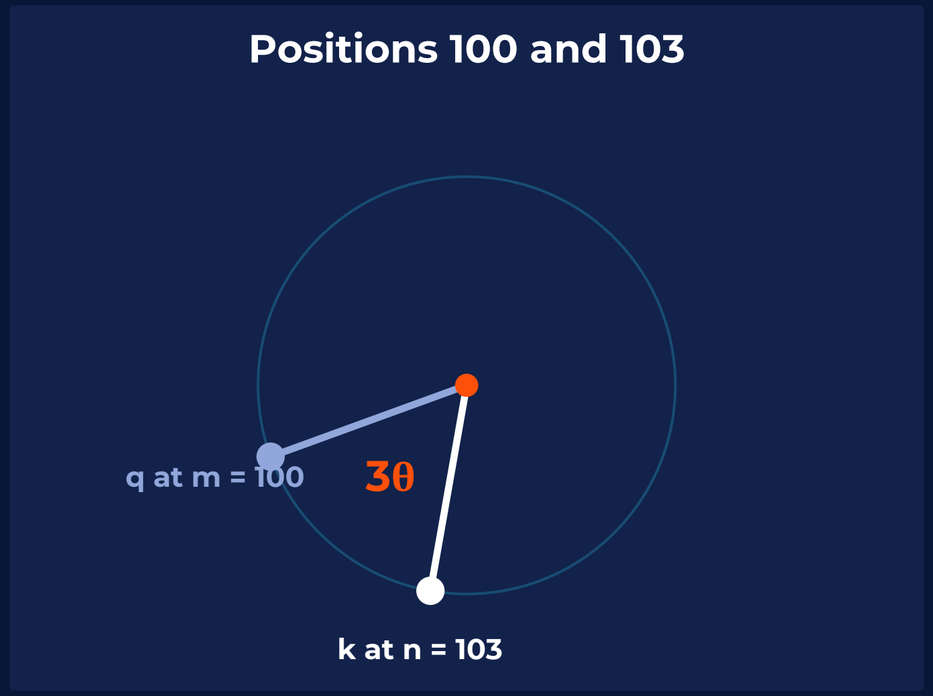
A partial fix exists: add a learned bias per distance to the logit (Shaw et al. 2018; T5). It buys relativity - but it puts a lookup inside the attention loop.

Shaw et al. (2018), Self-Attention with Relative Position Representations. NAACL 2018 · Raffel et al. (2020), T5. JMLR 21(140).

Rotate the query and the key

Same pair of vectors, two very different absolute positions





Both absolute angles changed by 98θ. The angle between them did not change at all.

Su et al. (2024), RoFormer: Enhanced Transformer with Rotary Position Embedding. Neurocomputing 568:127063.

Relative position

Absolute rotations applied separately pr oduce a relative-only inner product.

ROTATE EACH 2-D PAIR OF THE HEAD

$$R_m^{(i)} = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}$$

EACH PAIR SPINS AT ITS OWN RATE

$$\theta_i = b^{-2i/d}, \qquad i = 0, 1, \ldots, \tfrac{d}{2} - 1$$

$$R_m^\top R_m = I \;\Rightarrow\; \|R_m q\| = \|q\|$$

$$\langle R_m q,\ R_n k \rangle = q^\top R_{n-m}\, k$$

No parameters

Rotation angles are fixed by a formula, not learned

Nothing added to the logit

Applied to q and k before the dot product

Norm preserved

Rotations are orthogonal, so attention scale is untouched

Su et al. (2024), Neurocomputing 568:127063 (preprint arXiv:2104.09864, 2021).

One knob sets how far the model sees

$$\lambda_i = 2\pi\, b^{2i/d}$$

The slowest-rotating pair sets the longest distance the model can still distinguish. If we raise the base (b) and every wavelength stretches.

10,000

Original RoPE and Llama 2

4k context

500,000

Llama 3

8k trained, extends to 128k

10k / 1M

Gemma 3 — split by layer type

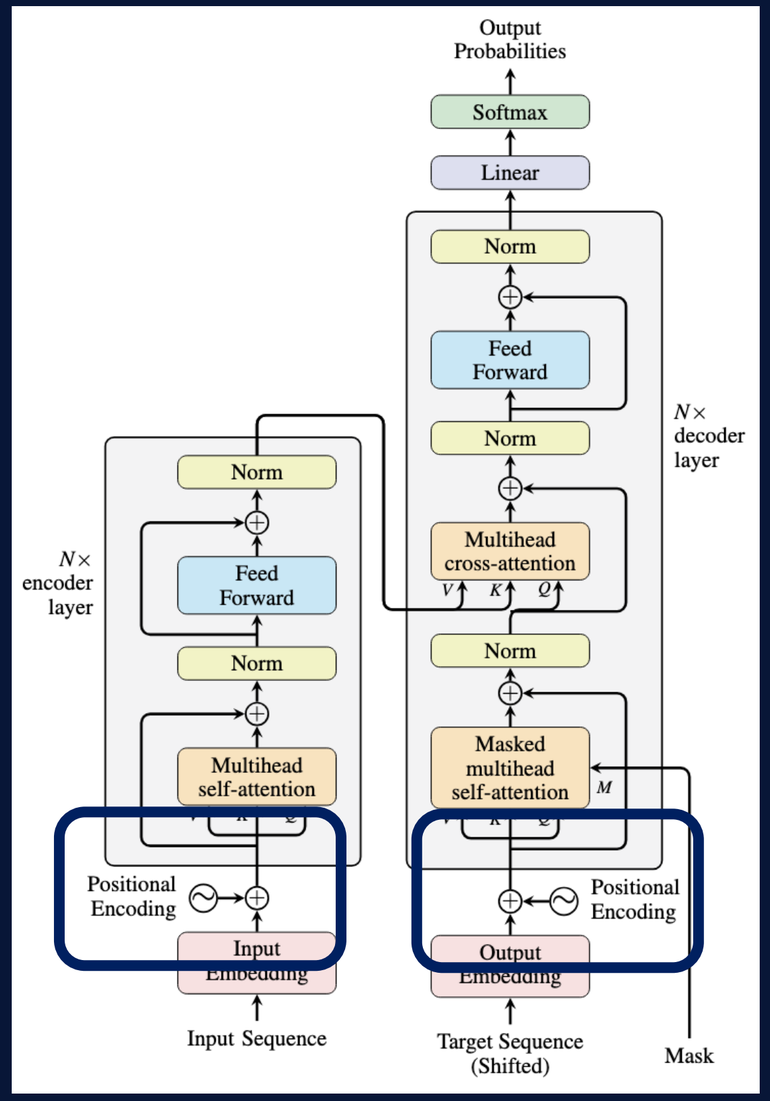
local layers 10k, global layers 1M

Gemma 3 runs two different bases in one model: fast rotation on local sliding-window layers, slow rotation on the global layers that carry long-range structure.

Grattafiori et al. (2024), The Llama 3 Herd of Models. arXiv:2407.21783 · Gemma Team (2025), Gemma 3 Technical Report.

03
Make decoding affordable

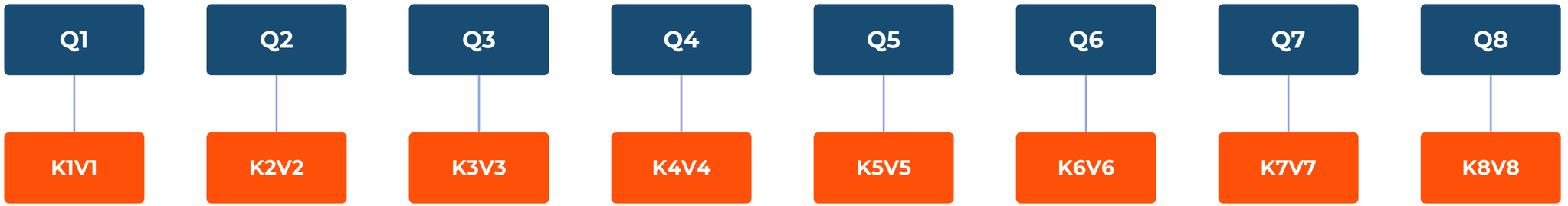
The only change today that is not about quality at all.



Each query head brings its own K and V

BEFORE | WHAT BREAKS | THE FIX

Multi-head attention, as discussed on Day 2



8 key/value heads - one cached pair per query head

$$\text{head}_j = \text{Attn}\left(QW_j^Q, KW_j^K, VW_j^V\right), \quad j = 1, \ldots, h$$

Fine while training. The problem starts at generation time.

During autoregressive decoding, every key and value computed so far must be kept - otherwise each new token would recompute the entire prefix.

Vaswani et al. (2017), NeurIPS 2017 · caching described in Shazeer (2019), arXiv:1911.02150.

The cache, not compute, is the bottleneck

Llama-3-8B shape: 32 layers, 128-dim heads, 8k context, fp16

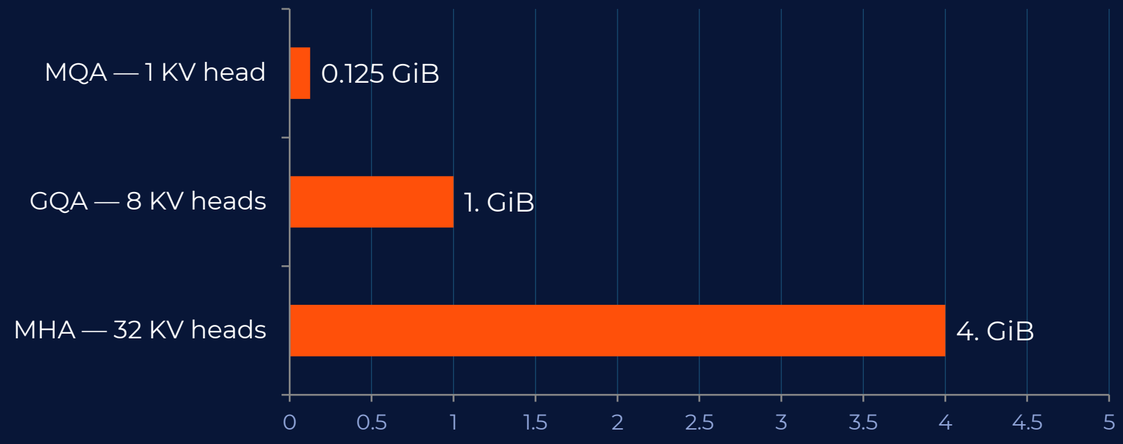
$$\text{Cache} = 2 \cdot B \cdot L \cdot T \cdot h_{kv} \cdot d_h \cdot s$$

2	B	L	T	h	d	s
keys and values	batch	layers	tokens so far	KV heads	head dim	bytes/elem

2 × 32 layers × 32 heads × 128 dims × 8,192 tokens × 2 bytes

= 4 GiB

for a single user, on a model whose weights are ~16 GiB



Grows with every token

The cache is not a fixed overhead — it expands as the answer gets longer.

Grows with every user

Batch 32 turns that 4 GiB into 128 GiB. This is why serving is hard.

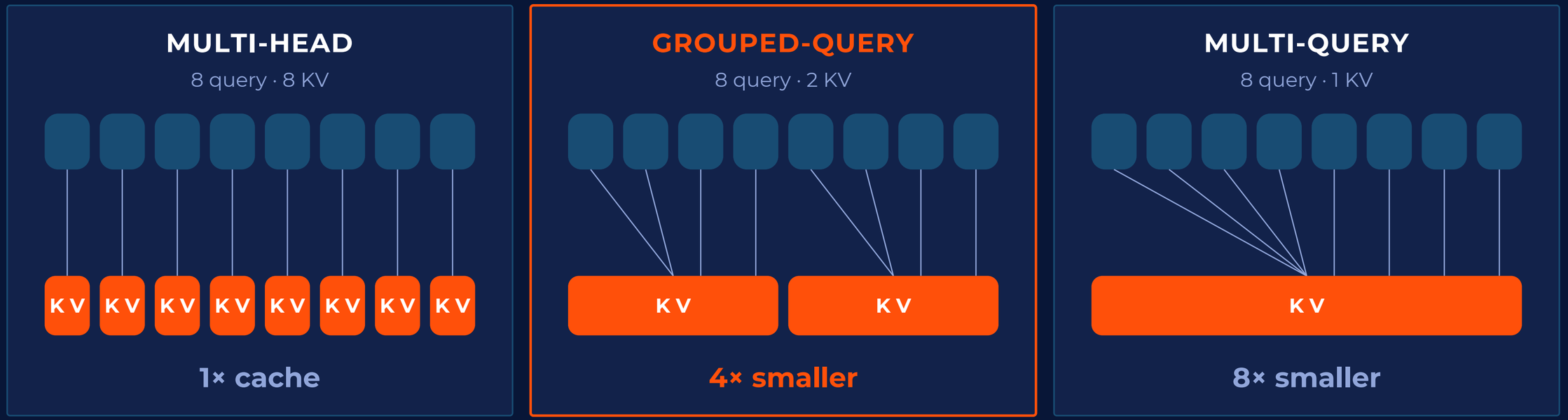
Decoding is memory-bound

One token's worth of arithmetic requires reading the whole cache from memory.

Cache formula standard; shapes from Grattafiori et al. (2024), arXiv:2407.21783. Single sequence, fp16.

Share keys and values across groups

GQA: with multi-head and multi-query as its two endpoints



$$h_{kv} = h_q \Rightarrow \text{MHA} \qquad h_{kv} = 1 \Rightarrow \text{MQA}$$

Llama 3 · Qwen 3 · Gemma 3 all use 8 key/value heads. The number of query heads varies; the KV count barely does.

Ainslie et al. (2023), GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP 2023 · Shazeer (2019), arXiv:1911.02150.

Mathing it!

The reduction is exactly linear

Cache size scales with key/value heads, so the ratio is the only thing you choose.

$$\frac{\text{Cache}_{\text{GQA}}}{\text{Cache}_{\text{MHA}}} = \frac{h_{kv}}{h_q} = \frac{1}{g}$$

UPTRAINING FROM MHA

$$W^K_{\text{group}} = \frac{1}{g} \sum_{j \in \text{group}} W^K_j$$

Llama 3 8B

32 → 8

query heads to KV heads

g = 4, cache quartered

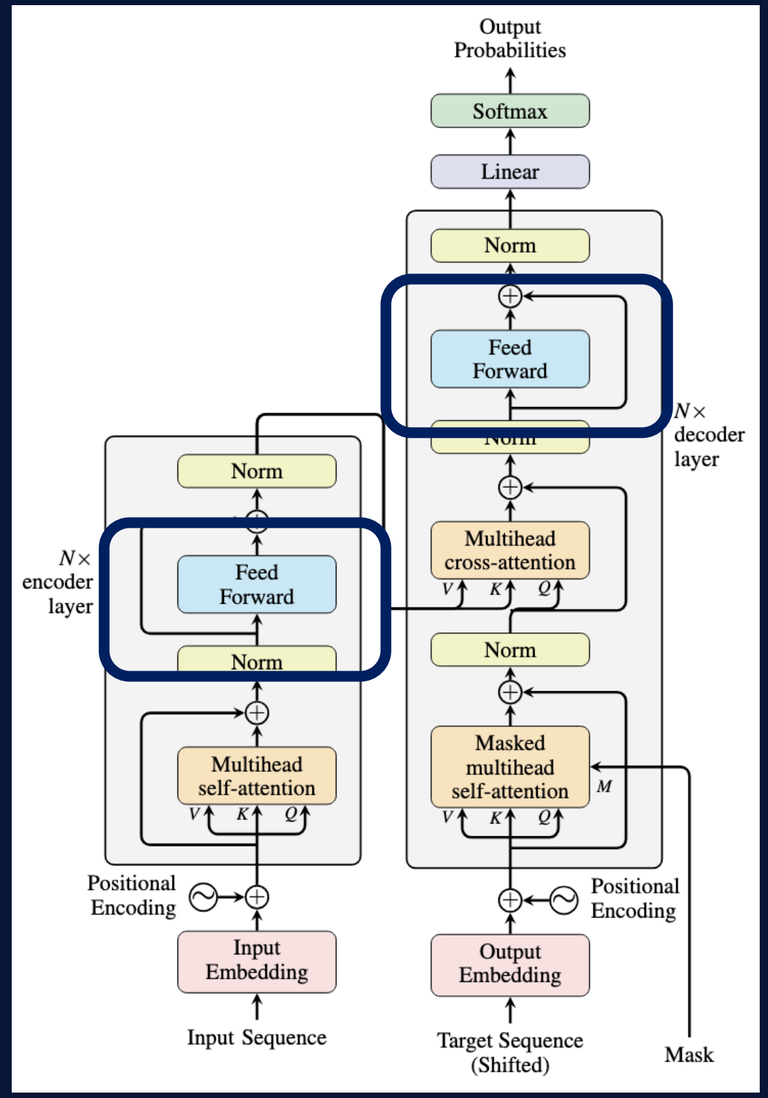
- Group size g is a genuine quality-versus-memory trade- but the curve is unusually flat near g = 4 to 8.
- Existing multi-head checkpoints can be converted by mean-pooling K and V within each group, then briefly continuing training.
- Attention FLOPs are almost unchanged, but memory traffic shrinks during decoding, which is what limits throughput.

Ainslie et al. (2023), EMNLP 2023.

04

Spend FFN parameters better

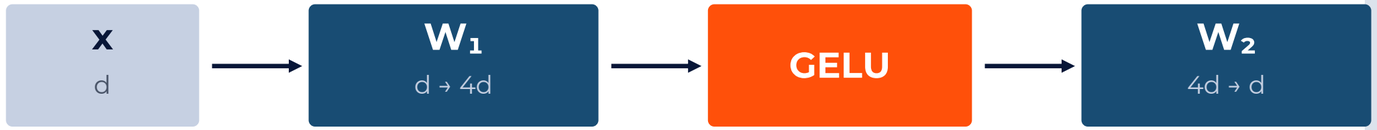
Two thirds of the block's parameters live here. Are they well used?



The feed-forward branch, unchanged

BEFORE | WHAT BREAKS | THE FIX

Widen, apply a pointwise nonlinearity, project back



Two weight matrices. Each feature is either passed or suppressed, independently of the others.

Share of block parameters

≈ 2 / 3

$8d^2$ in the feed-forward branch versus $4d^2$ in attention

$$\mathrm{FFN}(x) = \mathrm{GELU}(xW_1 + b_1)\,W_2 + b_2$$

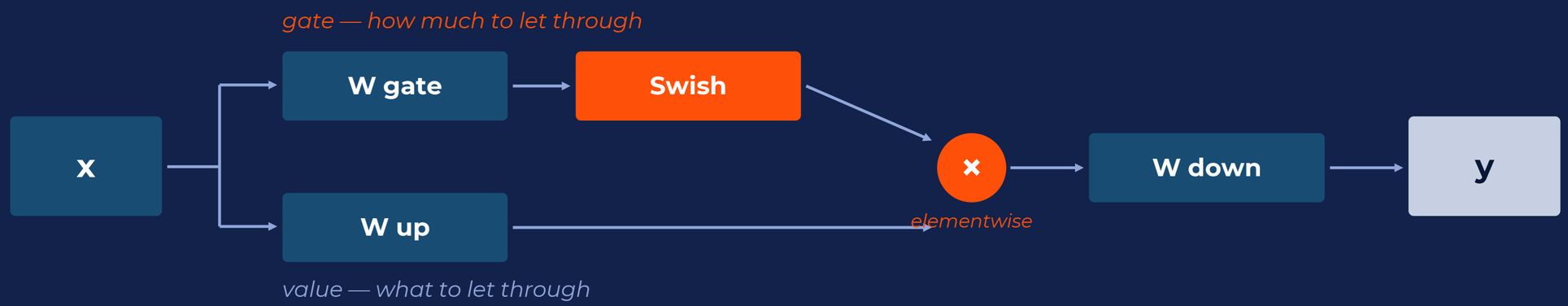
Most of your parameter budget, one nonlinearity

If this is where the parameters are, it is worth asking whether a pointwise activation is the best thing to do with them.

Vaswani et al. (2017), NeurIPS · Hendrycks & Gimpel (2016), Gaussian Error Linear Units. arXiv:1606.08415.

Let the layer decide what to let through

Two parallel projections, one gating the other



$$\text{Swish}_\beta(x) = x\,\sigma(\beta x)$$

Swish is smooth and non-monotonic

$$\text{GLU}(x) = \sigma(xW) \odot (xV)$$

Gating predates transformers — Dauphin et al., ICML 2017

The multiply is the whole idea: one projection now controls how much of another gets through.

Dauphin et al. (2017), ICML · Shazeer (2020), GLU Variants Improve Transformer. arXiv:2002.05202.

PAYING FOR THE THIRD MATRIX

Why the hidden width shrinks to 8d/3

Three matrices instead of two, so the width drops to keep the comparison fair.

$$\mathrm{FFN}_{\mathrm{SwiGLU}}(x) = \left(\mathrm{Swish}_1(xW_g) \odot xW_u\right)W_d$$

$$3\,d\,d_{ff}^{\text{gated}} = 2\,d\,d_{ff}^{\text{MLP}} \implies d_{ff}^{\text{gated}} = \frac{2}{3}\,d_{ff}^{\text{MLP}}$$

Match parameter count, then compare quality — otherwise you are just measuring size.

Llama 3 8B rounds this to 14,336 for hardware alignment

What the evidence says

At matched parameters and FLOPs, gated variants gave consistently better perplexity across Shazeer's transfer-learning suite.

What it does not say

No theoretical account of why gating helps. The original paper is explicit that the result is empirical.

What everyone did anyway

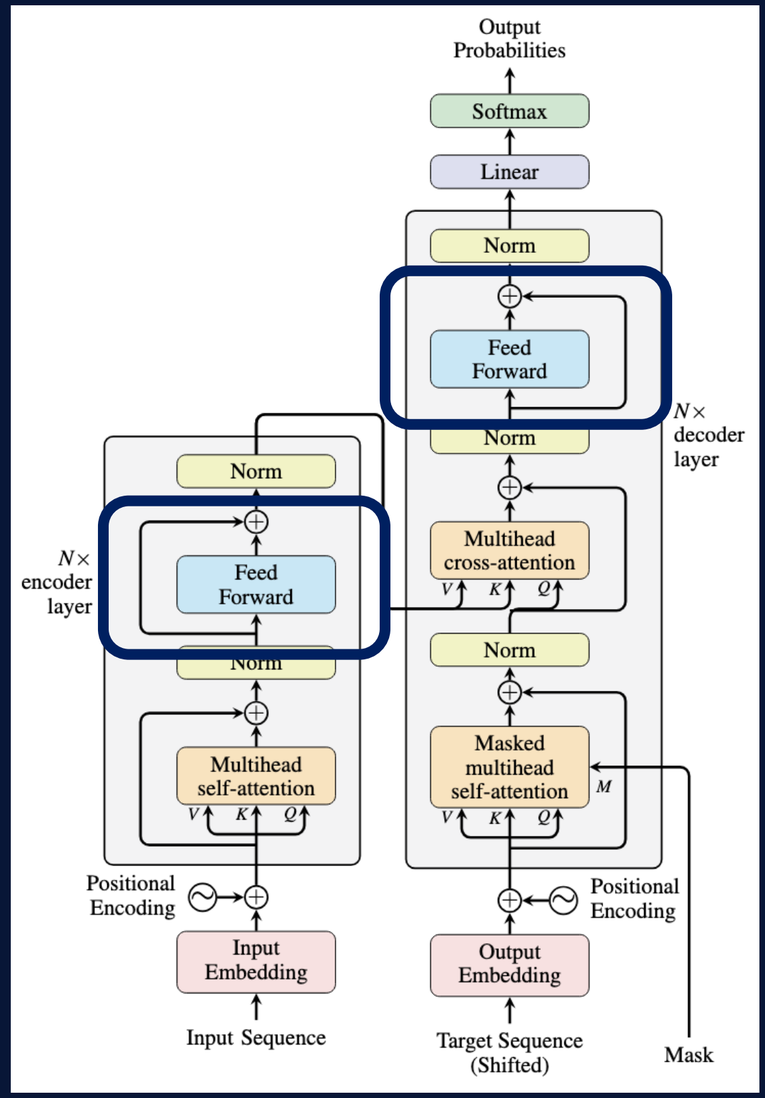
Llama, Mistral, Qwen and most open models adopted SwiGLU. Gemma 3 uses a gated GELU — same structure, different squashing function.

Shazeer (2020), arXiv:2002.05202 · Grattafiori et al. (2024), arXiv:2407.21783 · Gemma Team (2025), Gemma 3 Technical Report.

05

MoE: Decouple capacity from compute

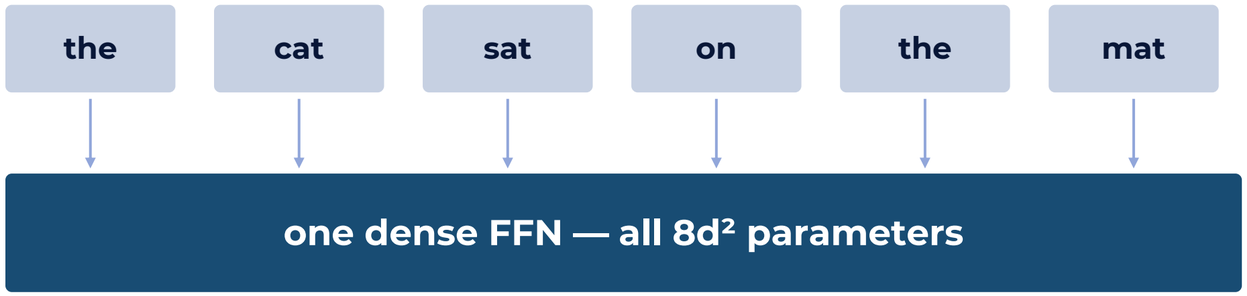
The only upgrade today that changes what a parameter count means.



Every token pays for every parameter

BEFORE | WHAT BREAKS | THE FIX

The dense feed-forward branch, as it stands



Identical compute per token, whatever the token is.

THE SCALING TENSION

More parameters
→ better models

More parameters
→ more FLOPs per token
In a dense model these are the same axis.
Capacity and cost cannot be separated.

$$y = \mathrm{FFN}(x)$$

A function word and a rare technical term get exactly the same computation.

That uniformity is the assumption mixture-of-experts drops.

Kaplan et al. (2020), arXiv:2001.08361 · Hoffmann et al. (2022), Training Compute-Optimal Large Language Models. NeurIPS 2022.

Route each token to a few experts

One router, N expert FFNs, k of them active per token

MIXTRAL 8×7B

8
experts per layer

2
active per token

46.7B
total parameters

13B
active parameters

Shazeer et al. (2017), Outrageously Large Neural Networks. ICLR 2017 · Jiang et al. (2024), Mixtral of Experts. arXiv:2401.04088.

ROUTING

Function of the layer

Score every expert, keep the best k, renormalize, combine.

SELECT

$$\mathrm{TopK}(v)_i = \begin{cases} v_i & v_i \in \text{top-}k \\ -\infty & \text{otherwise} \end{cases}$$

WEIGHT, THEN COMBINE

$$G(x) = \mathrm{softmax}\big(\mathrm{TopK}(x\,W_r)\big)$$

$$y = \sum_{i \in \mathcal{T}_k(x)} G(x)_i \cdot E_i(x)$$

$$P_{\text{total}} \propto N \qquad P_{\text{active}} \propto k \qquad k \ll N$$

Top-k is not differentiable

Gradients flow through the gate weights of selected experts only. Unselected experts get no signal from this token.

k is usually 1 or 2

Switch Transformer showed k = 1 works and simplifies routing. Mixtral uses k = 2.

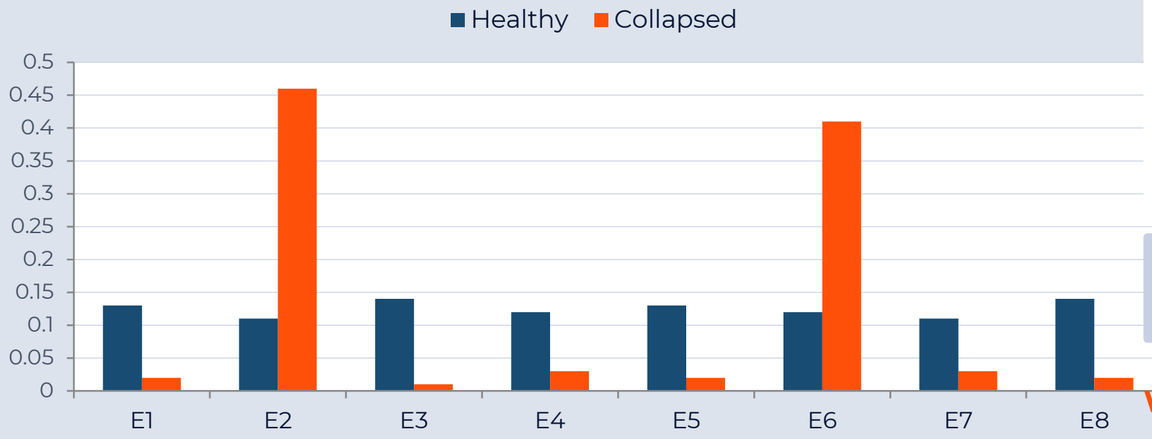
Softmax after selection

Renormalizing over the chosen k keeps the combination a convex sum.

Shazeer et al. (2017), ICLR · Fedus et al. (2022), Switch Transformers. JMLR 23(120) · Jiang et al. (2024), arXiv:2401.04088.

Routers collapse without a counterweight

Popularity is self-reinforcing: a chosen expert trains faster and gets chosen more.



Share of tokens routed to each expert

THE AUXILIARY LOSS

$$\mathcal{L}_{\text{aux}} = \alpha \, N \sum_{i=1}^{N} f_i \, P_i$$

f_i = token fraction to i $\quad$ P_i = mean gate prob.

Why it collapses

Only chosen experts receive gradient, so early luck compounds into permanent dominance.

What the loss does

It is minimised when tokens spread evenly, penalising the product of token share and gate probability.

The cost of ignoring it

Most experts sit unused — you pay full memory for capacity that never trains.

Shazeer et al. (2017), ICLR · Lepikhin et al. (2021), GShard. ICLR 2021 · Fedus et al. (2022), JMLR 23(120). Distributions illustrative.

MoE buys compute with memory

WHAT YOU GAIN

- Far more parameters at fixed per-token FLOPs
- Better quality per unit of training compute
- Experts shard naturally across devices

WHAT YOU PAY

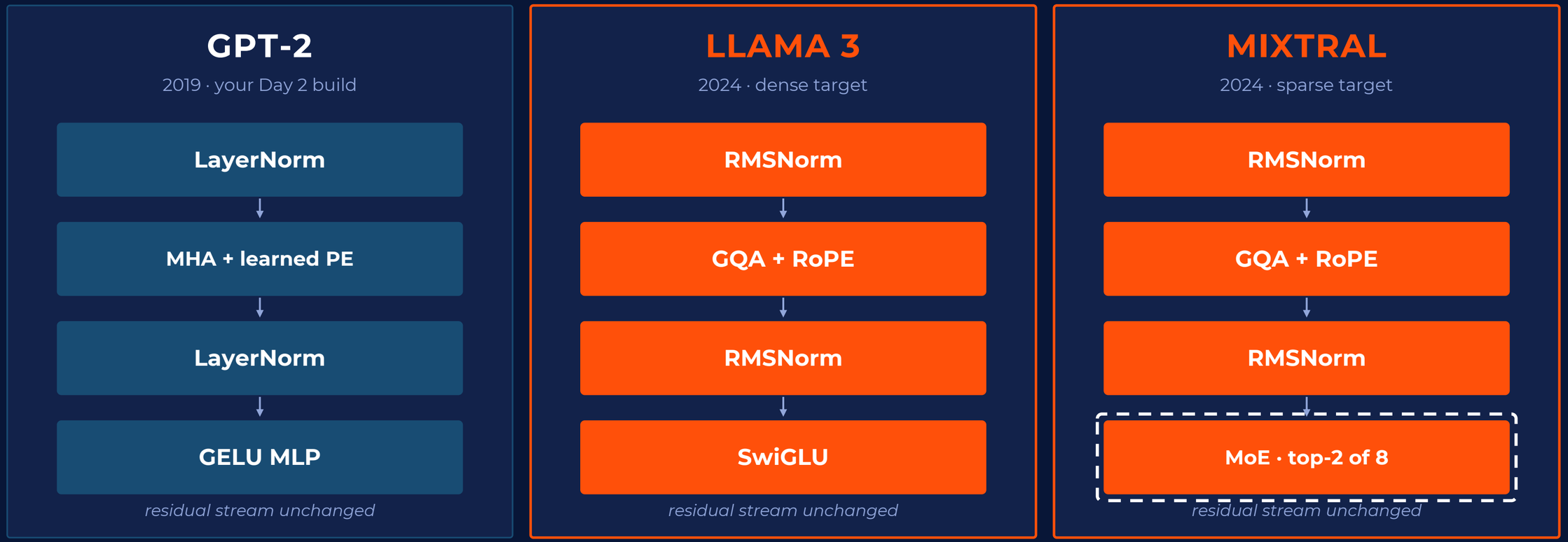
- Every expert must sit in memory, whether used or not
- Routing adds communication and load-balancing complexity
- Small batches leave most experts idle — poor utilisation

For an edge-deployed SLM, memory is usually the binding constraint — not FLOPs.

A 47B-parameter sparse model needs 47B parameters' worth of memory to serve one user. If your target device has 8 GB, the compute saving is irrelevant. Build dense, measure, and only reach for MoE when you have shown that compute — not memory — is what limits you.

Jiang et al. (2024), arXiv:2401.04088 — the paper notes serving memory scales with total, not active, parameters.

The same block, in Frontier Models



Mixtral differs from Llama 3 in exactly one component. That is the whole of sparsity.

READING A TECHNICAL REPORT

Three families, one recipe

Once you know the five slots, any new model is a short list of differences.

	LLAMA 3	QWEN 3	GEMMA 3
Normalization	RMSNorm, pre-norm	RMSNorm + QK-norm	RMSNorm, pre and post
Position	RoPE, base 500k	RoPE	RoPE, 10k local / 1M global
Attention	GQA, 8 KV heads	GQA, no QKV bias	GQA, 5:1 local:global
Feed-forward	SwiGLU	SwiGLU	Gated GELU
Sparsity	Dense only	Dense and MoE, 8 of 128	Dense only

Every cell is one of the five slots from this session. No new mechanisms — only different settings.

Grattafiori et al. (2024), arXiv:2407.21783 · Yang et al. (2025), arXiv:2505.09388 · Gemma Team (2025), Gemma 3 Technical Report.

HANDS-ON

Just ponder upon Day 2 block!

One component at a time, with a test after each. The block interface never changes.

ORDER OF WORK

1. RMSNorm, then verify scale invariance
2. RoPE on q and k only — never on values
3. GQA, then check it reduces to your Day 2 output
4. SwiGLU at 8d/3 width
5. Swap in the MoE layer last

THE FIVE INVARIANTS

1. Output shape equals input shape
2. Future tokens cannot affect earlier outputs
3. Gradients reach every parameter
4. Parameter count matches your budget
5. Every expert receives some tokens

GO DEEPER

References

Primary sources for everything claimed today, cited at the venue where each appeared.

NORMALIZATION

Vaswani et al. (2017). Attention Is All You Need. NeurIPS.

Ba, Kiros & Hinton (2016). Layer Normalization. arXiv:1607.06450.

Xiong et al. (2020). On Layer Normalization in the Transformer Architecture. ICML.

Zhang & Sennrich (2019). Root Mean Square Layer Normalization. NeurIPS.

POSITION

Shaw et al. (2018). Self-Attention with Relative Position Representations. NAACL.

Raffel et al. (2020). Exploring the Limits of Transfer Learning (T5). JMLR 21(140).

Su et al. (2024). RoFormer. Neurocomputing 568:127063.

Press et al. (2022). Train Short, Test Long (ALiBi). ICLR.

ATTENTION & FFN

Shazeer (2019). Fast Transformer Decoding (MQA). arXiv:1911.02150.

Ainslie et al. (2023). GQA. EMNLP.

Dauphin et al. (2017). Gated Convolutional Networks. ICML.

Shazeer (2020). GLU Variants Improve Transformer. arXiv:2002.05202.

SPARSITY & SYSTEMS

Shazeer et al. (2017). Outrageously Large Neural Networks. ICLR.

Lepikhin et al. (2021). GShard. ICLR.

Fedus et al. (2022). Switch Transformers. JMLR 23(120).

Jiang et al. (2024). Mixtral of Experts. arXiv:2401.04088.

Start here — Xiong et al. (2020) and Ainslie et al. (2023). Both are short, and each shows a failure diagnosed and fixed.

Questions!

IAIRO
Indian AI Research Organisation

Thank you

SPEAKER

Mohit Joshi

mhtjsh.github.io

MATERIALS

https://iairobootcamp.zite.so/

Slides, notebook and reading list

NEXT SESSION

Day 4 · Data Engineering

BPE tokenizers, filtering, deduplication