Architecture for open models

Shreya Pathak

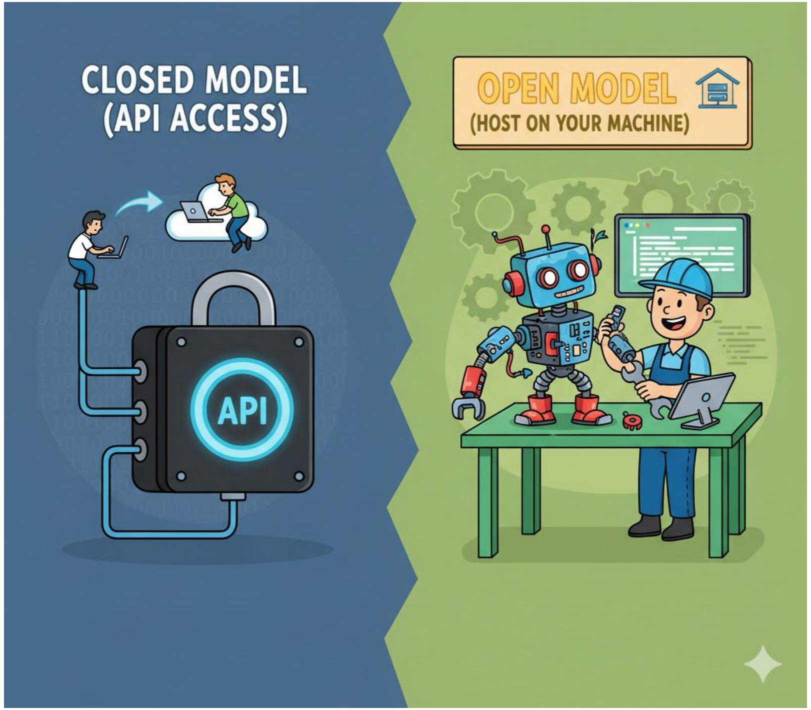
Google DeepMind

ML in PL

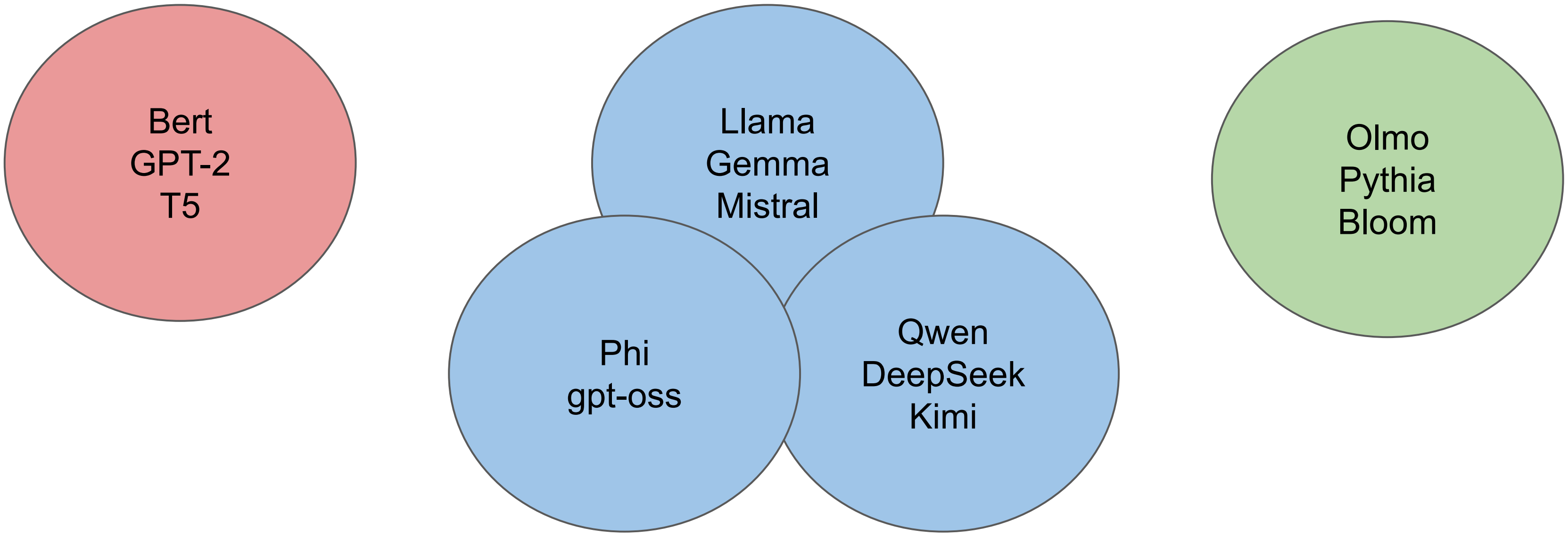
CONFERENCE 2025

Why open / on-device models?

- Deploy on own hardware
- Scale as needed
- No data transfer
  - Privacy
  - No bandwidth latency
- Offline usage
- Can be finetuned



Popular open models



Ingredients of pre-training an LLM

- **Architecture**
- Data
- Optimization
- Quantization
- Modalities / Capabilities
  - Vision
  - Long-context
- Evaluation

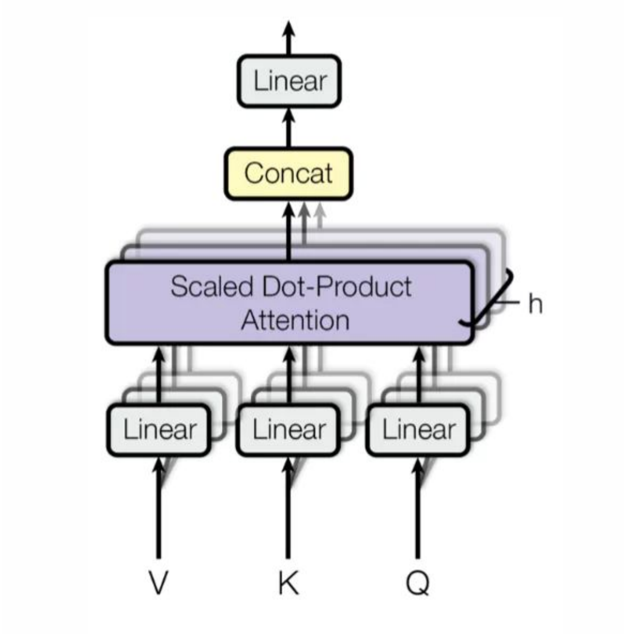
Aim of architecture choices in on device models

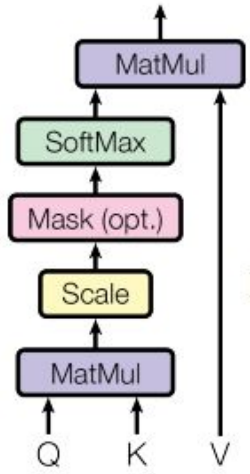
- Improve accuracy
  - Eg, better residual connection
- Reduce latency
  - Eg, Smaller model
- Reduce KV cache
  - Eg, MLA
- Make model easier to quantize
  - Eg, more norms
- Increase stability
  - Eg, more norms
- Simplicity
  - Good for open source

How to evaluate architecture changes?

- Perplexity on different domain data
  - Cleaner so more signal
  - Downstreams only used for larger runs
- Infra metrics
  - Inference latency
  - Training step time
  - Device memory usage
- Scalability
  - Change should hold across model sizes
- Stability
  - Gradient norms look normal, i.e., no spikes

Transformer basics



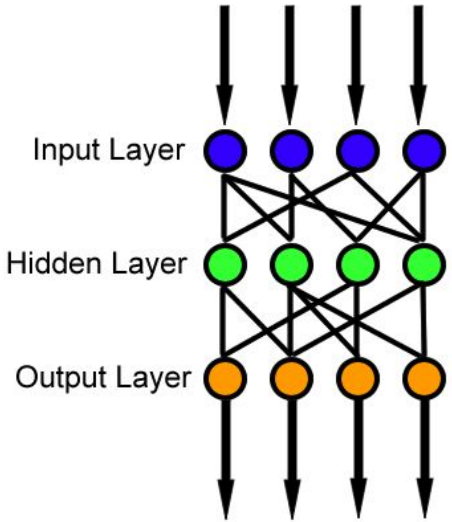


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Attention mechanism

Transformer basics



$$\frac{\text{attention FLOPs}}{\text{matmul FLOPs}} = \frac{12BT^2NH}{18BTDF + 24BTDNH} = \frac{12BT^2D}{4 * 18BTD^2 + 24BTD^2} = \frac{12BT^2D}{96BTD^2} = \frac{T}{8D}$$

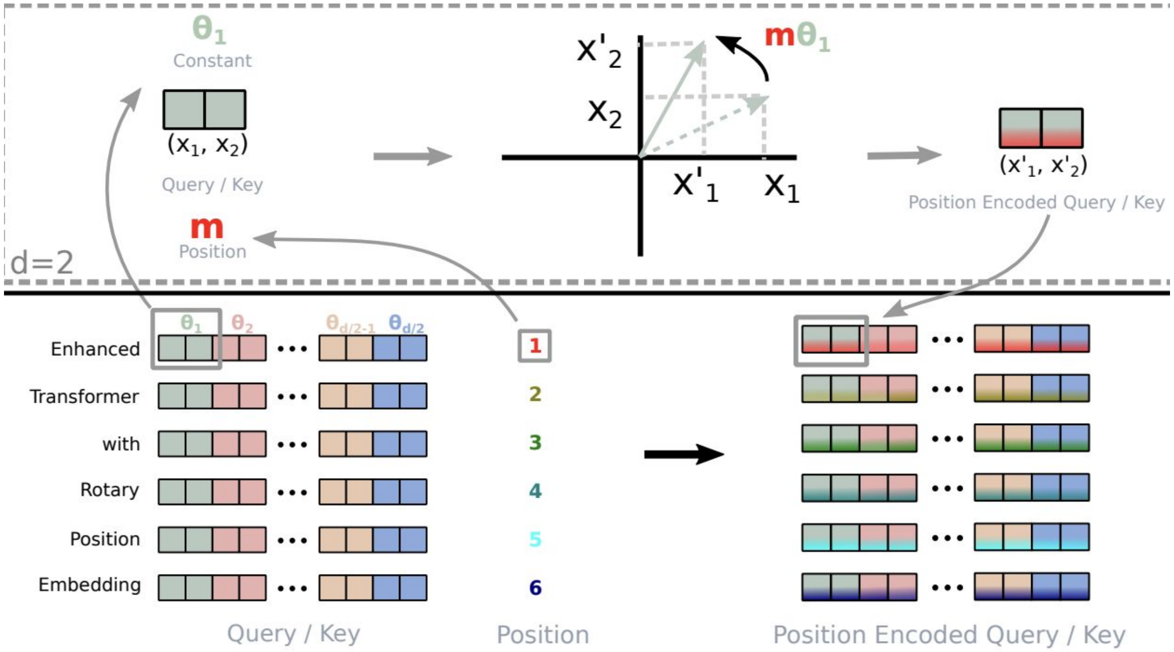
Feed Forward Layer

Wide and shallow

Deep and narrow

Aspect ratio

Transformer basics (cont.)



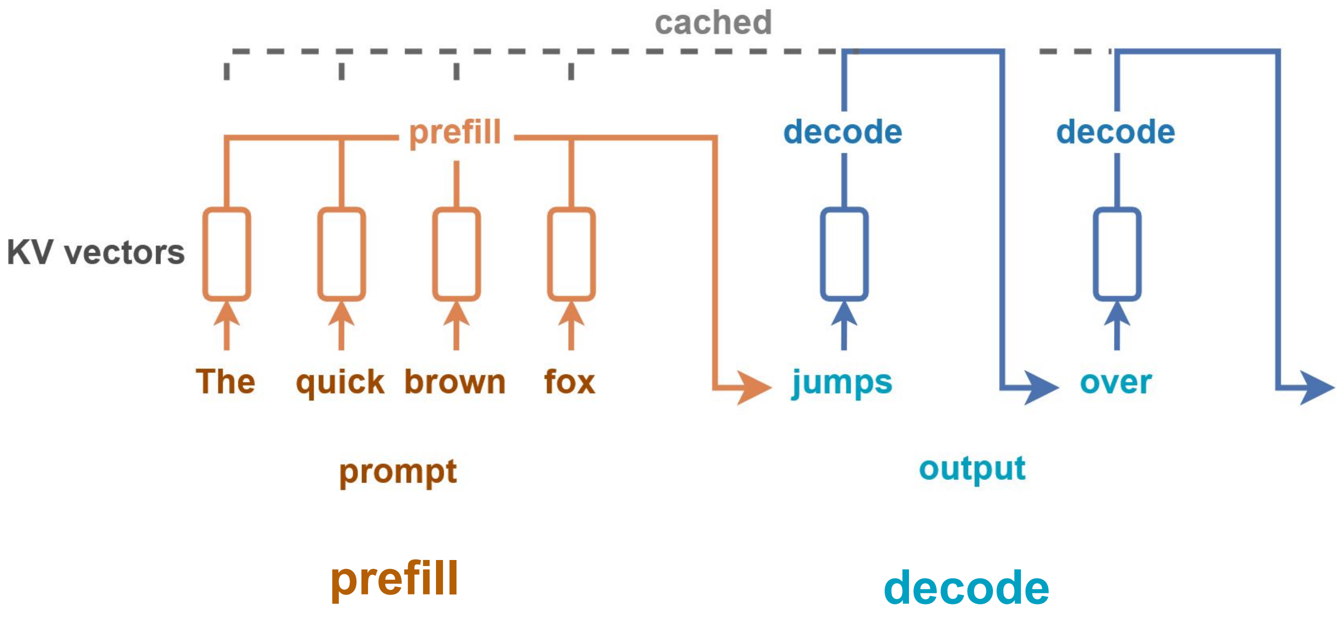
Hyperparams

- Wavelength
- Scale factor

Benefits:

- Relative distance
- Efficient
- Generalizable

Transformer basics (cont.)



KV cache is shared between prefill and decode

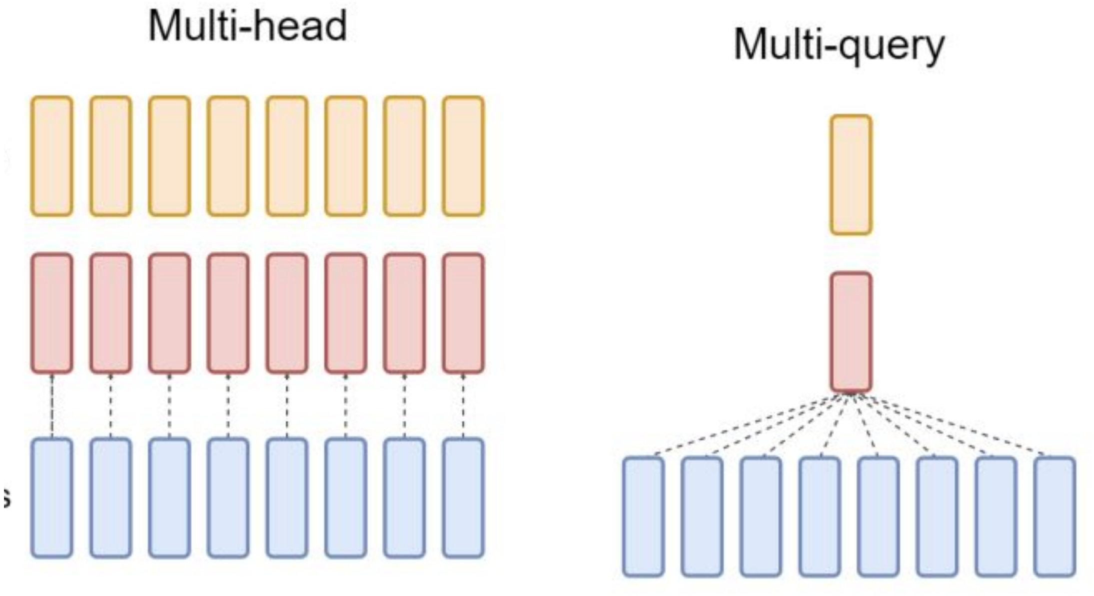
For a single layer and a single token

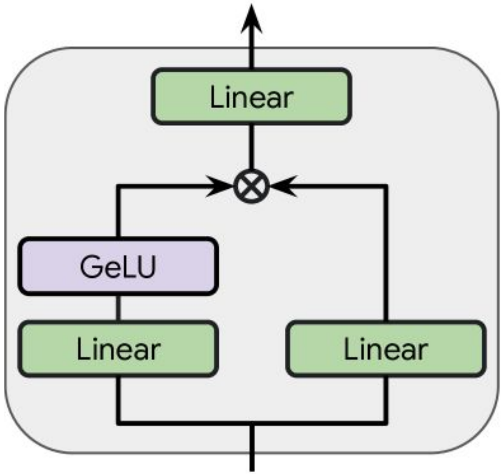
Size of KV cache =
$n_kv * seq\_len * head\_dim$

Gemma series of models

Rank (UB)	Model	Score	95% CI (±)	Votes	Organization	License
59	gemma-3-27b-it	1362	±4	43,379	Google	Gemma
67	gemma-3-12b-it	1340	±10	3,866	Google	Gemma
94	gemma-3n-e4b-it	1318	±5	23,755	Google	Gemma
111	gemma-3-4b-it	1302	±9	4,195	Google	Gemma
131	gemma-2-27b-it	1285	±3	76,195	Google	Gemma lice...
133	gemma-2-9b-it-simpo	1277	±7	10,108	Princeton	MIT
148	gemma-2-9b-it	1262	±4	54,954	Google	Gemma lice...
183	gemma-2-2b-it	1196	±4	46,901	Google	Gemma lice...
192	gemma-1.1-7b-it	1177	±6	24,327	Google	Gemma lice...

Gemma 1



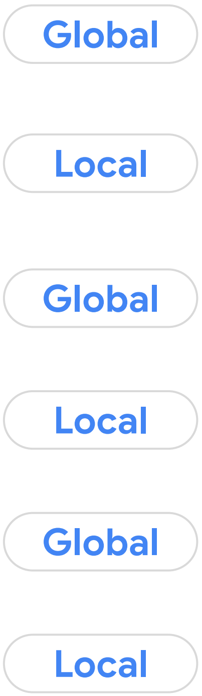


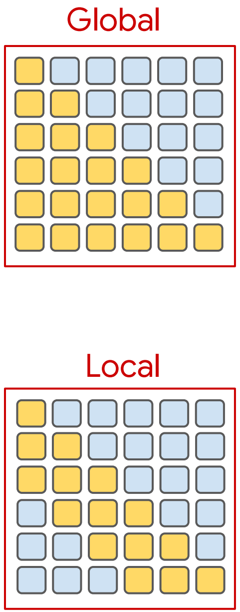
(b) Gated MLP block

https://arxiv.org/pdf/2403.08295

https://arxiv.org/pdf/2002.05202

Gemma 2





- 1:1 local global attention
- 4096 sliding window size

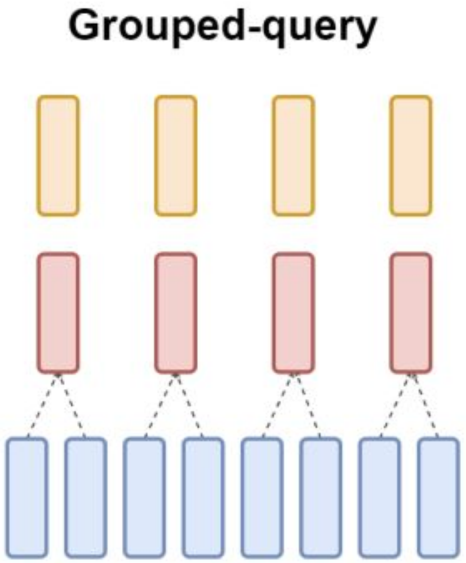
For a sequence length of 8192

All global kv cache,
$$C = L * (n * 8192 * h)$$

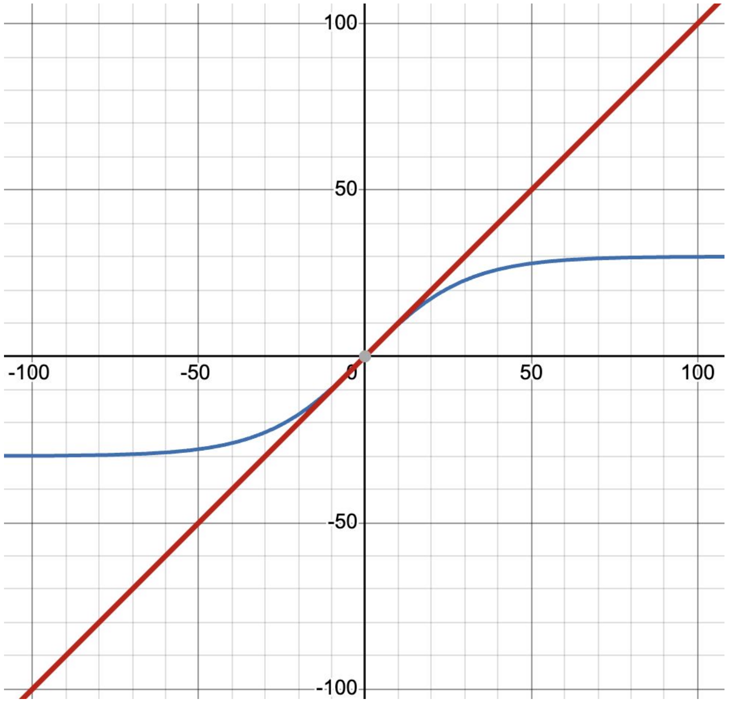
Interleaved kv cache =
$$C_i = \frac{L}{2} * (n * 4096 * h + n * 8192 * h)$$
$$= \frac{L}{2} * n * h * 12288 = \frac{3}{4} C$$

https://arxiv.org/pdf/2408.00118

Gemma 2 (cont)



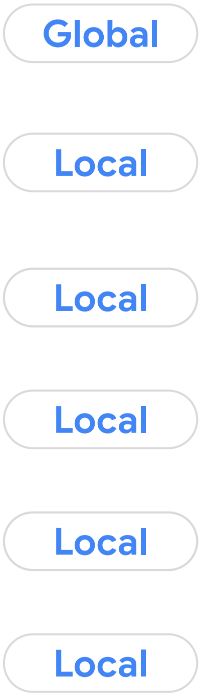
$$n_{kv} = \frac{n}{2}$$

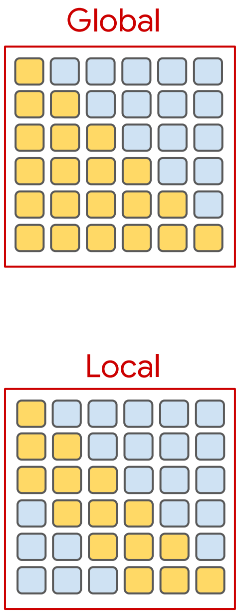


Softcapping: $y = \left(t \cdot \tanh\left(\frac{x}{t}\right)\right)$

https://arxiv.org/pdf/2408.00118

Gemma 3





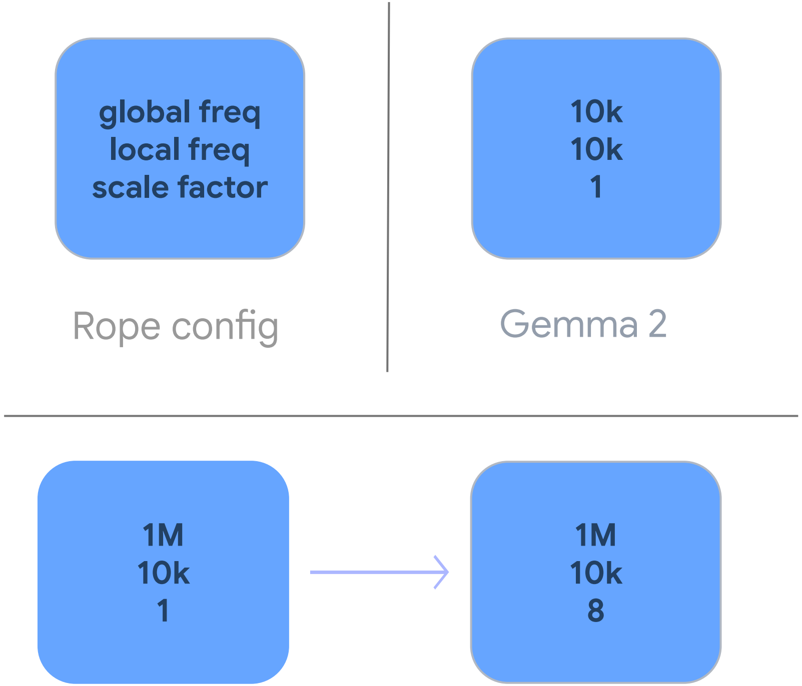
- 5:1 local global attention
- 1024 sliding window size

KV cache size =

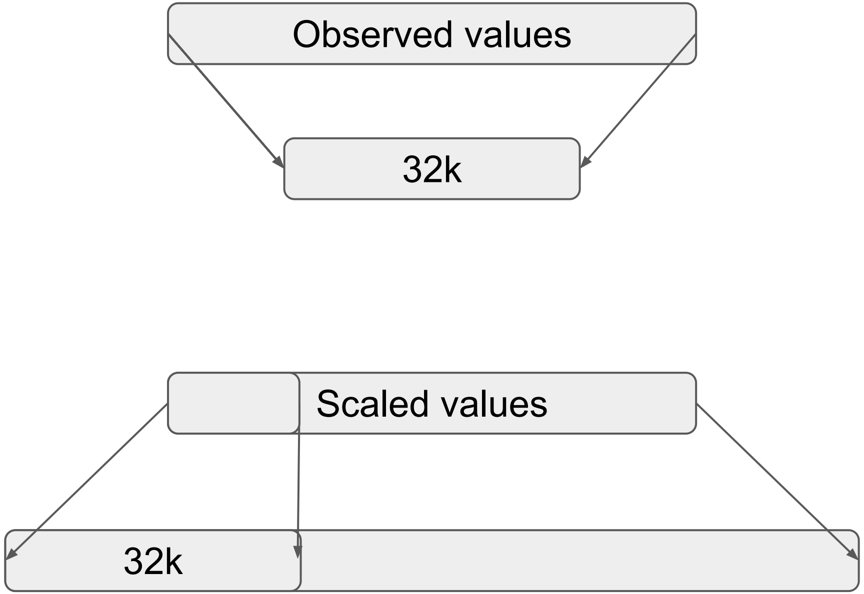
$$C_3 = \frac{L}{6} * (5 * n * 1024 * h + n * 8192 * h)$$
$$= \frac{L}{6} * n * h * 13312 = 0.27 * C$$

https://arxiv.org/pdf/2503.19786

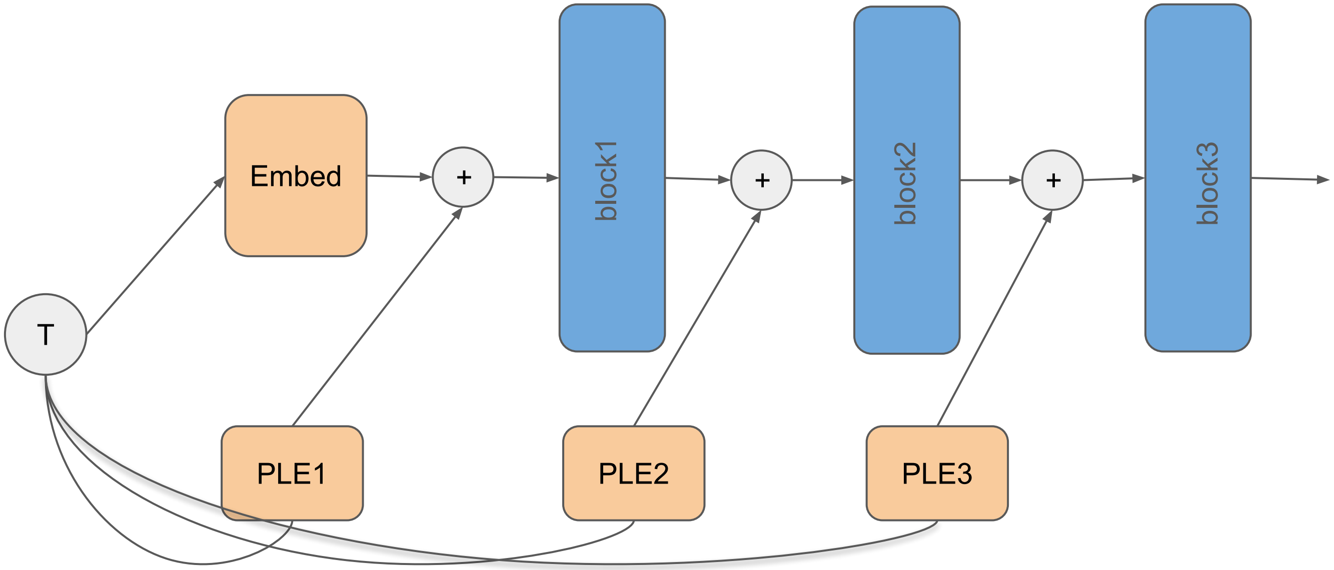
Gemma 3 (cont)



Gemma 3
Position interpolation



Gemma 3n

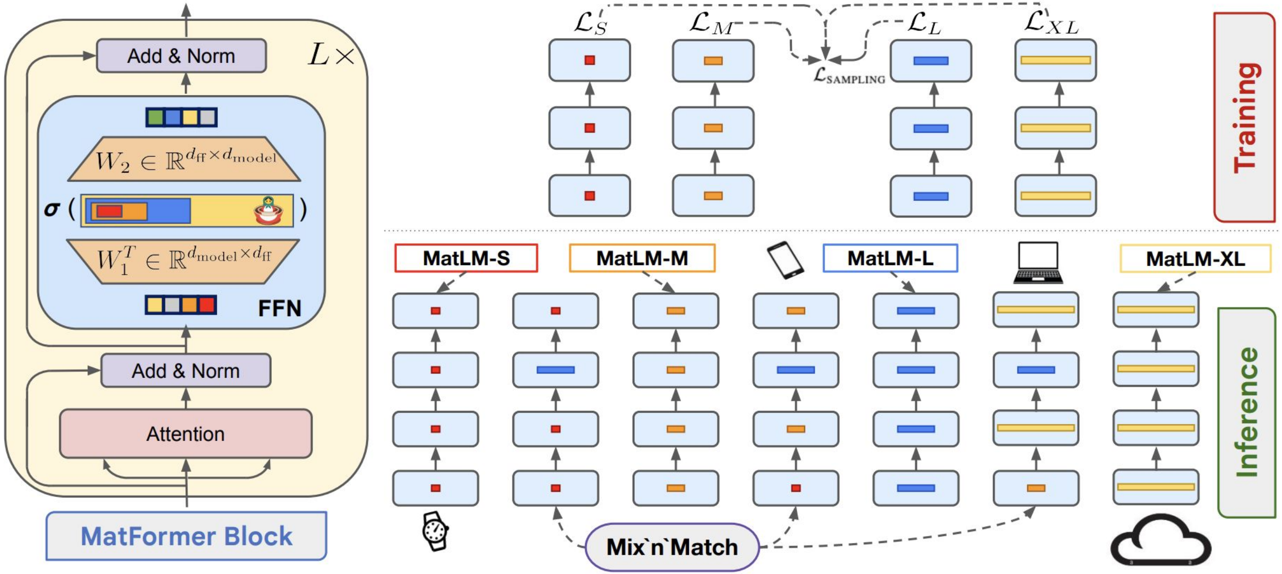


Per Layer Embeddings

- Improved accuracy
- Total trainable params increases
- PLE doesn't need to reside in device memory
- Cached in RAM

https://ai.google.dev/gemma/docs/gemma-3n

Gemma 3n (cont)

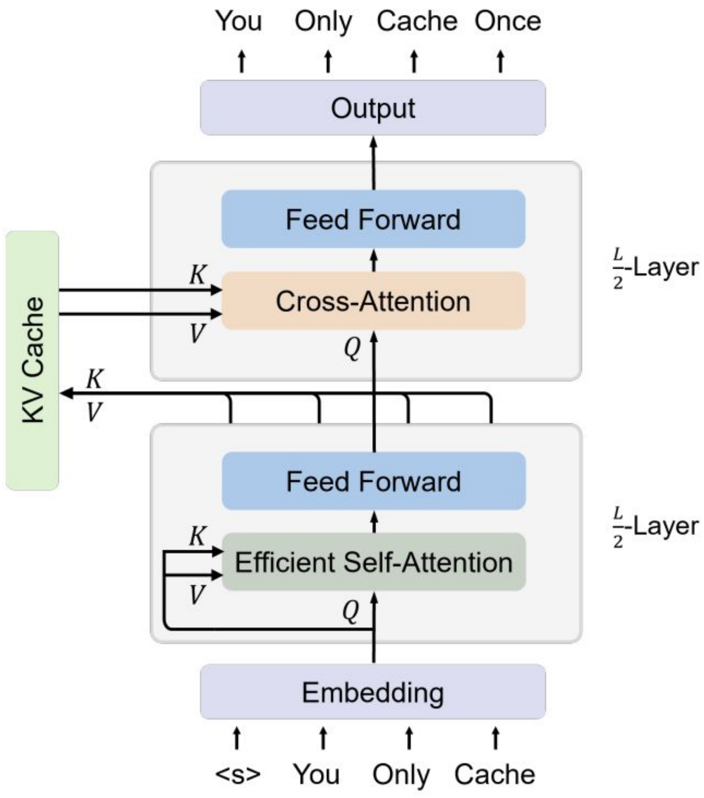


MatFormers in Gemma3n

- Nested submodels
- Applied only to FFN layers
- A single submodel for high latency requirement use cases

https://arxiv.org/pdf/2310.07707

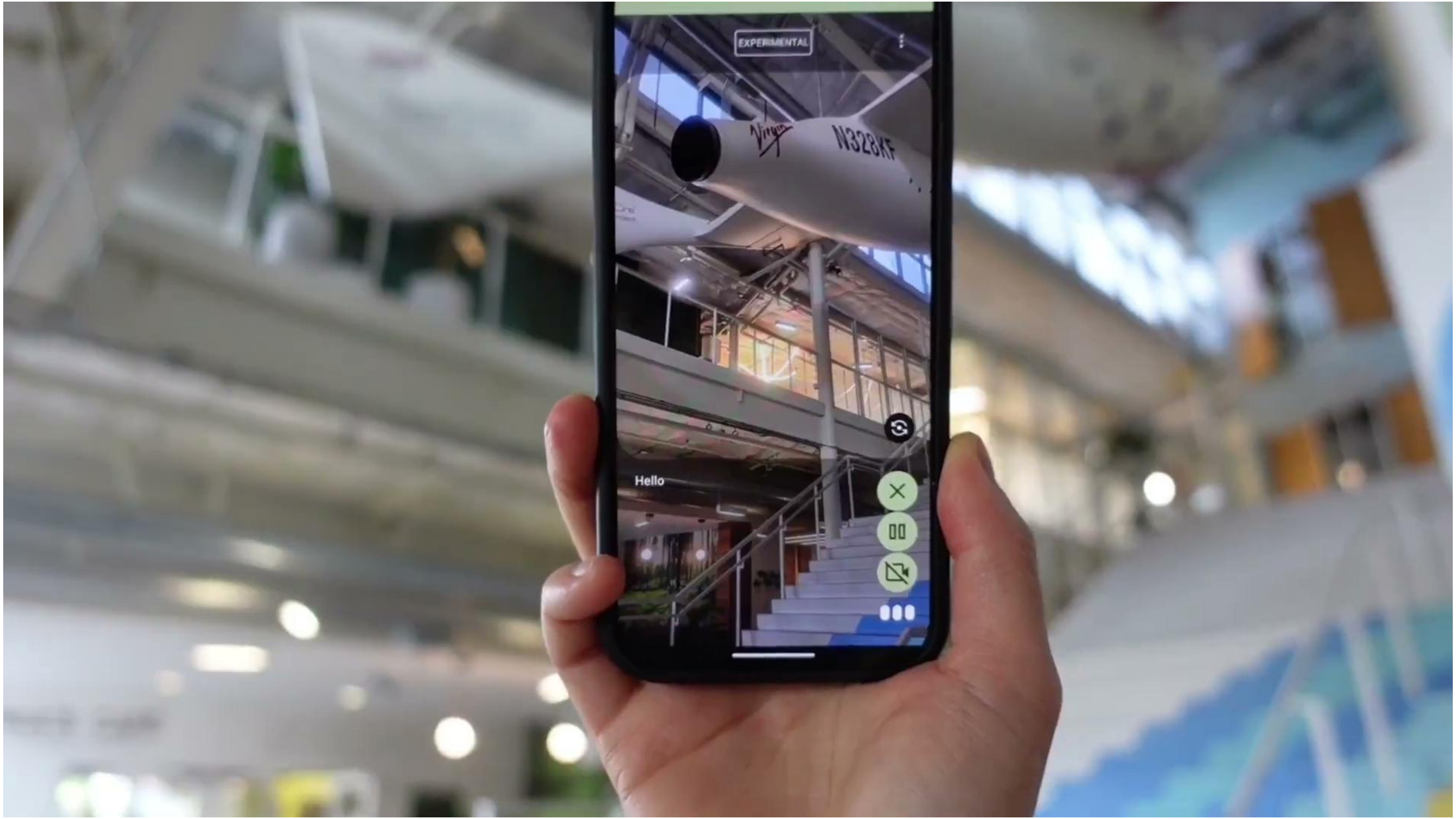
Gemma 3n (cont)



KV Sharing

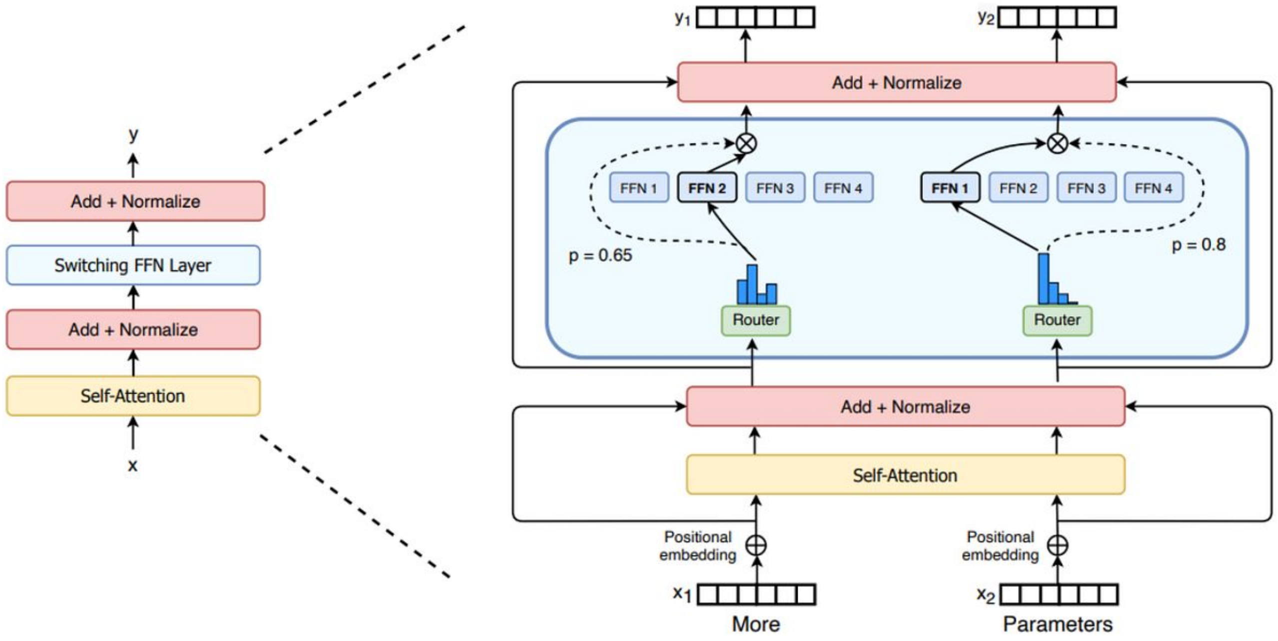
- First half of the layers calculate KV cache.
- Rest use the cache from the last layer.
- Prefill cost reduced to half

https://arxiv.org/pdf/2405.05254


EXPERIMENTAL
N328KF
Hello

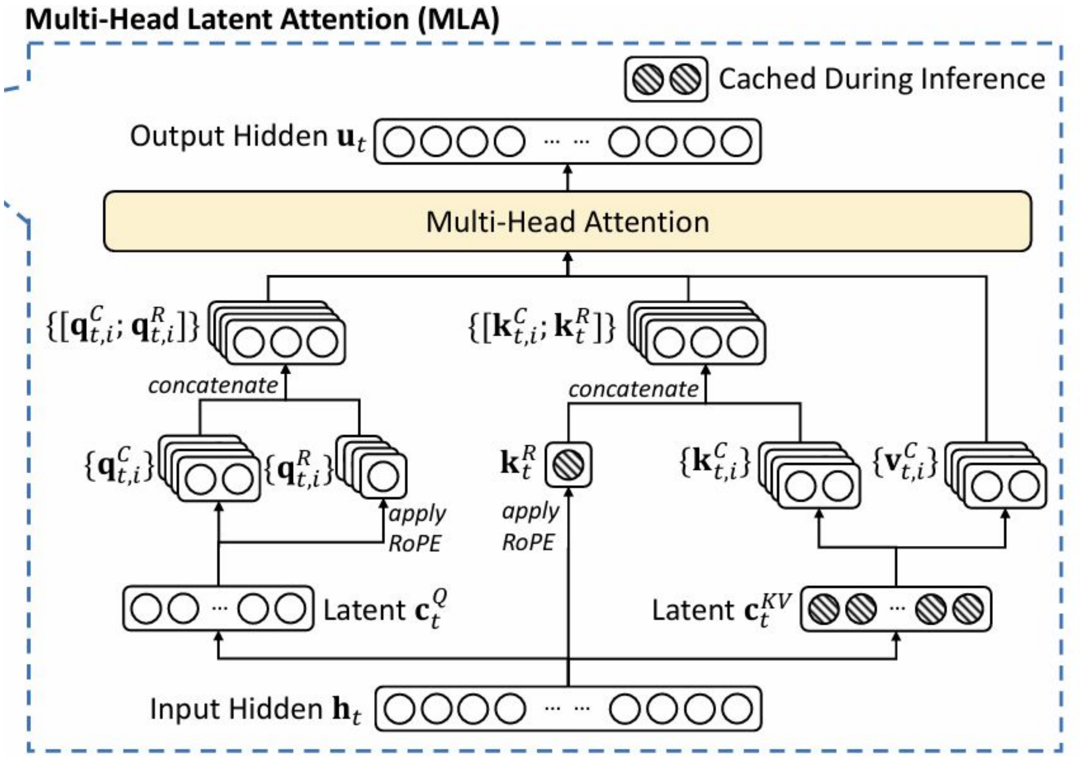
Interesting architecture updates from other open models.

Mixture of experts



- Improved accuracy at same flops
- Increased memory requirements
- Lower latency
- Load-balanced experts

Multi-head Latent Attention



- Significant KV cache savings while maintaining accuracy
- c_t^{KV}, k_t^R are cached

KV cache =

$$L * seq\_len * (latent\_dim + rope\_dim)$$

we get kv cache savings, if

$$latent\_dim + rope\_dim < 2 * (n * h)$$

https://arxiv.org/pdf/2405.04434

Thanks!