Deepseek-V2: Multi-Head Latent Attention

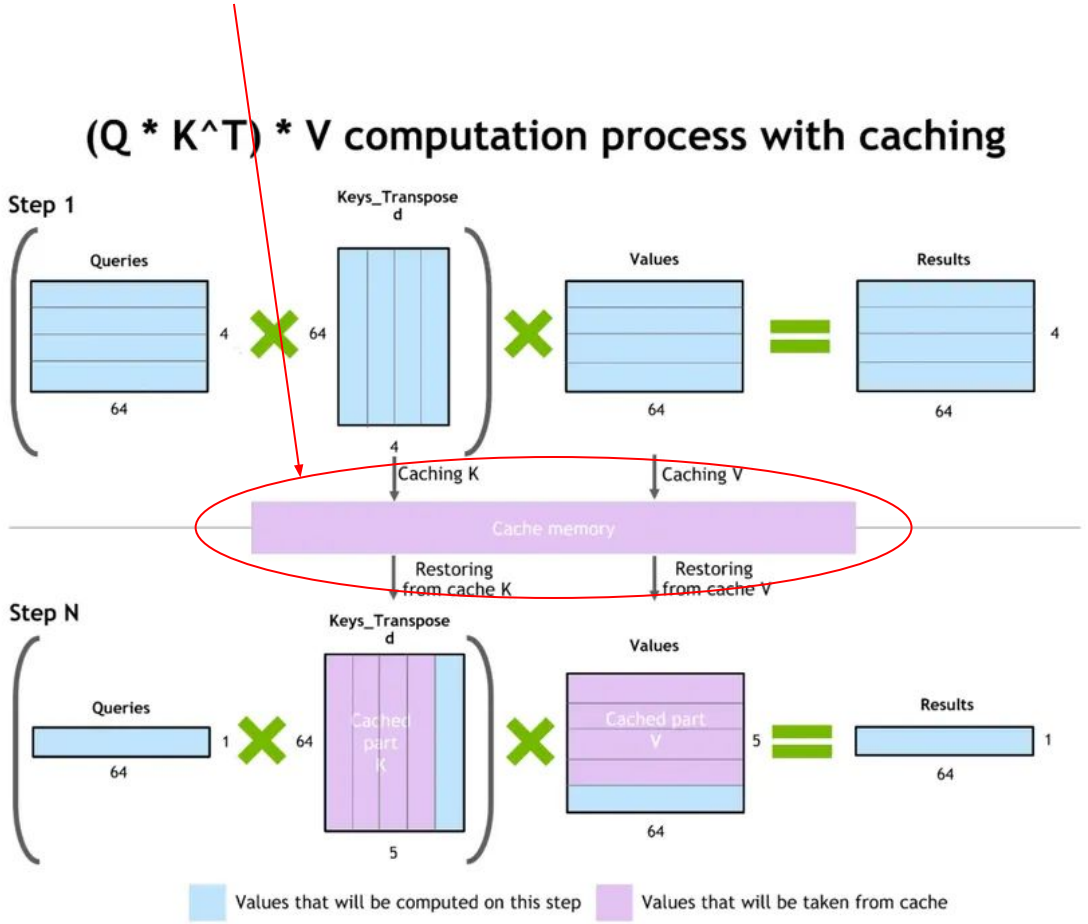
Vineet Kulkarni, Yashkaran Chouhan

Georgia Tech

Multi-Head Latent Attention: Making inference cache-friendly

- KV cache enables fast autoregressive decoding, but dominates memory traffic
- Inference cost grows with context length, not model size
- Multi-Head Latent Attention targets this bottleneck directly

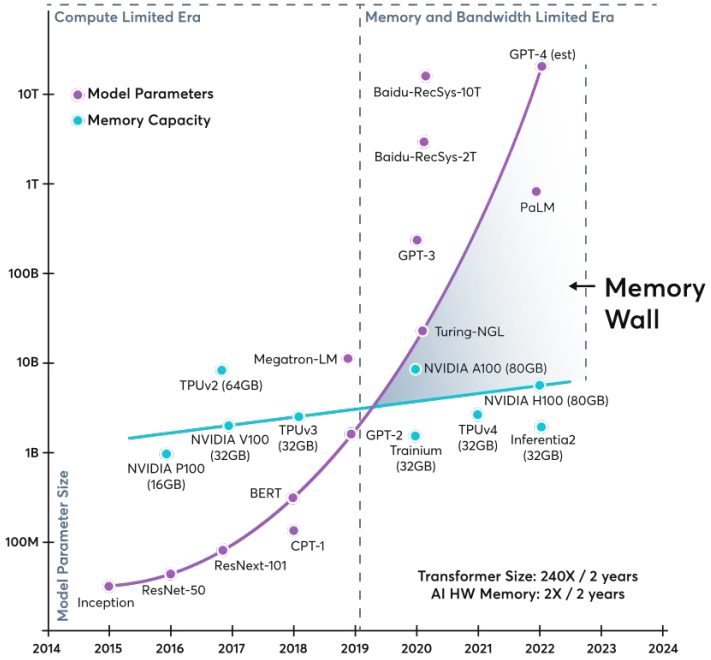
Inference bottleneck



Georgia Tech

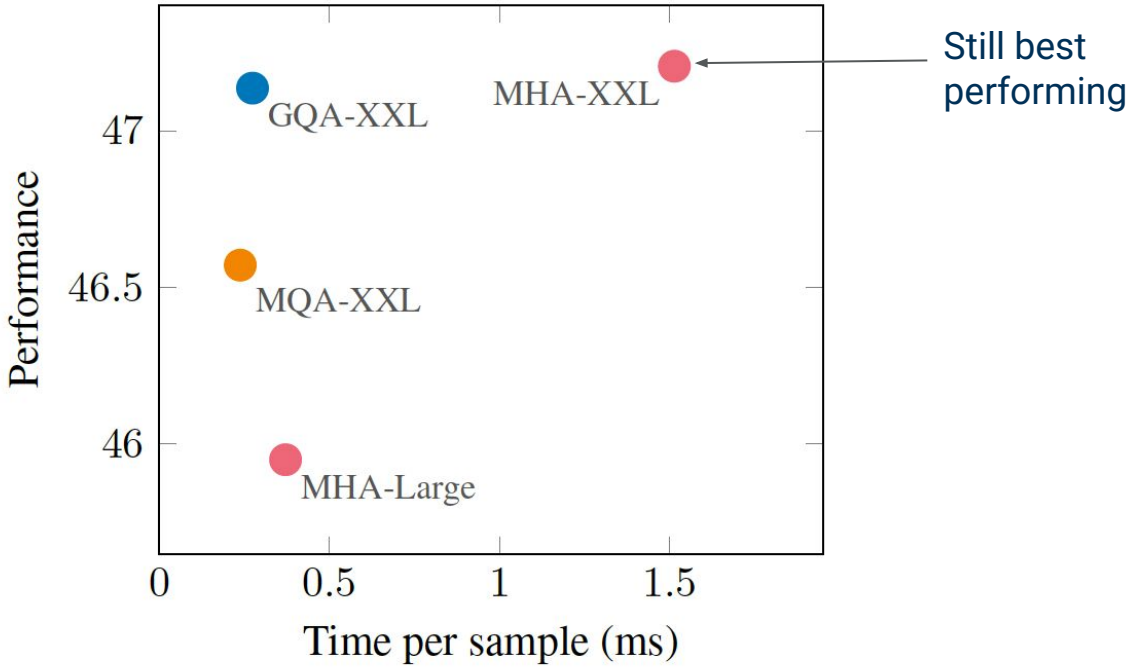
Why do we care about optimizing KV cache?

- LLM inference is increasingly memory bandwidth bound, not compute-bound
- KV Cache $\propto$ S (context length)
- Especially an issue with long-context inference
- Biggest problem: KV cache dominates cost and latency

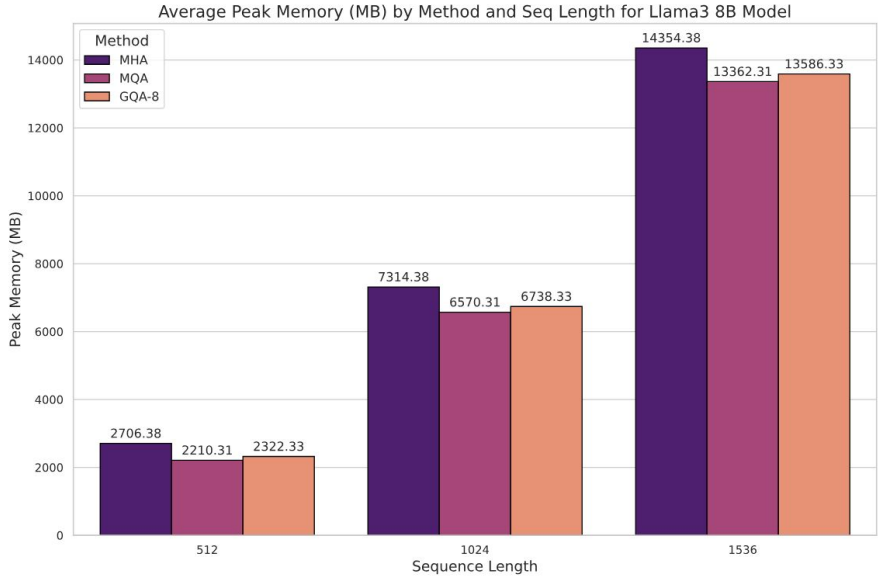


Key question: How do we reduce KV cache memory and bandwidth *without sacrificing MHA-level performance?*

- Must work with autoregressive decoding
- Must preserve attention expressivity
- Must scale to long context

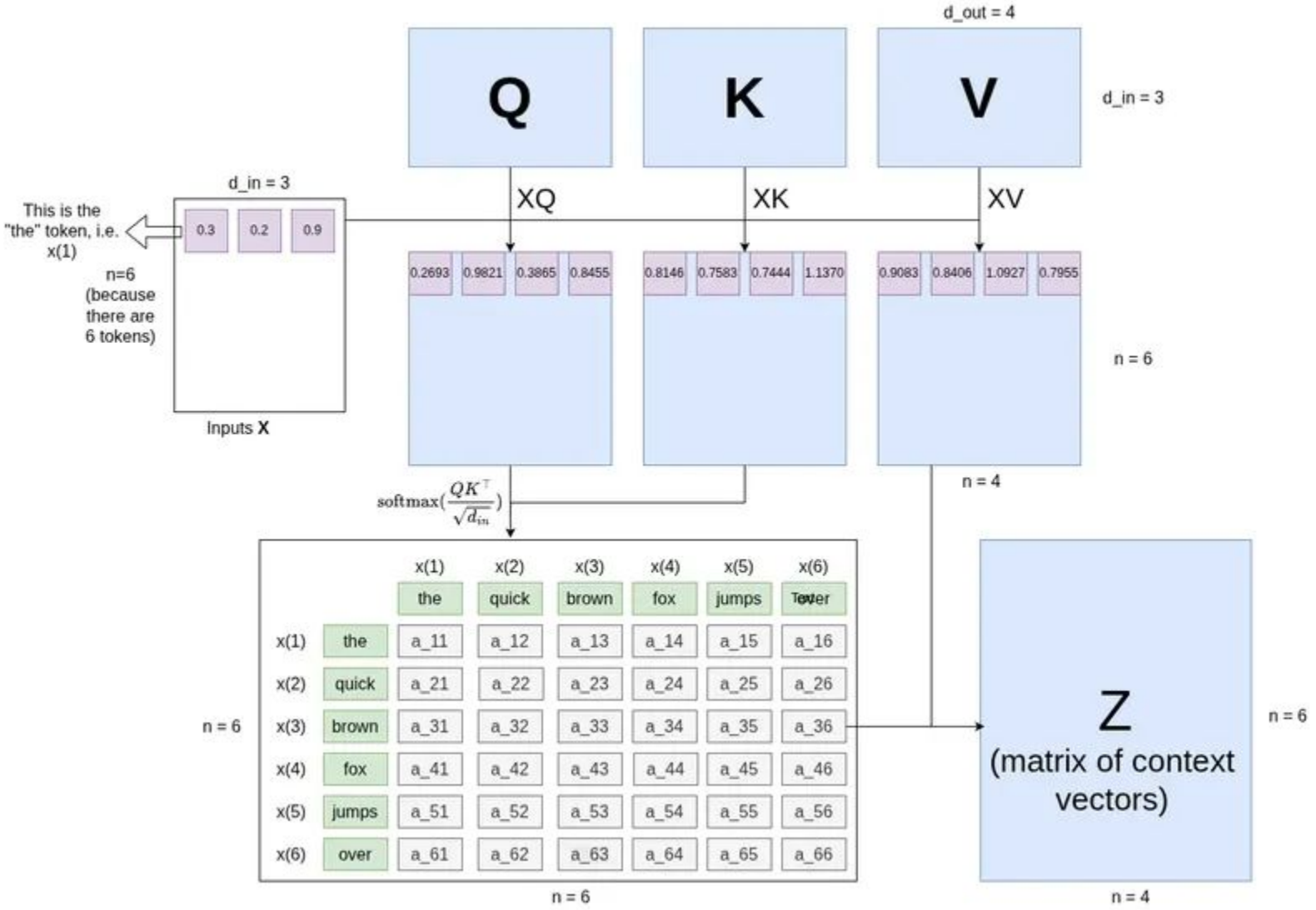


Still best performing

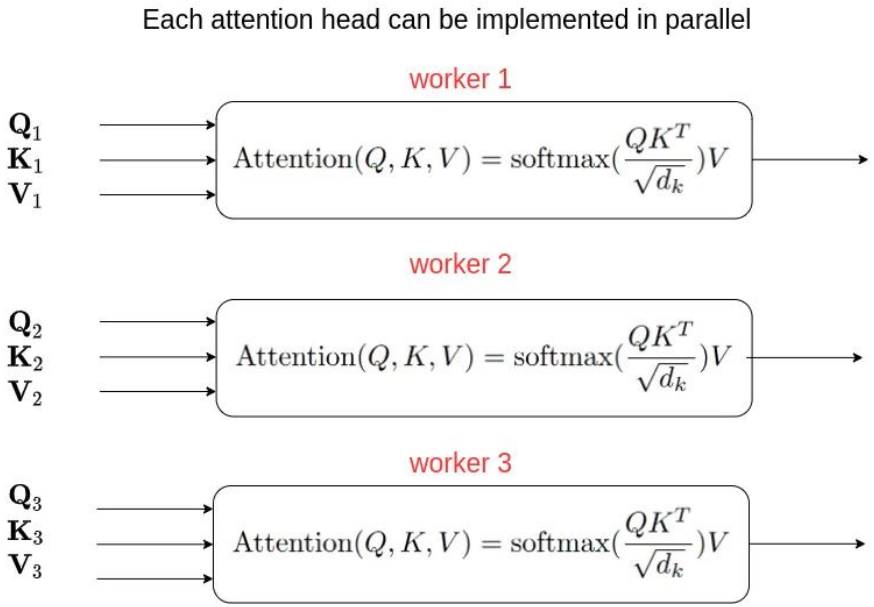


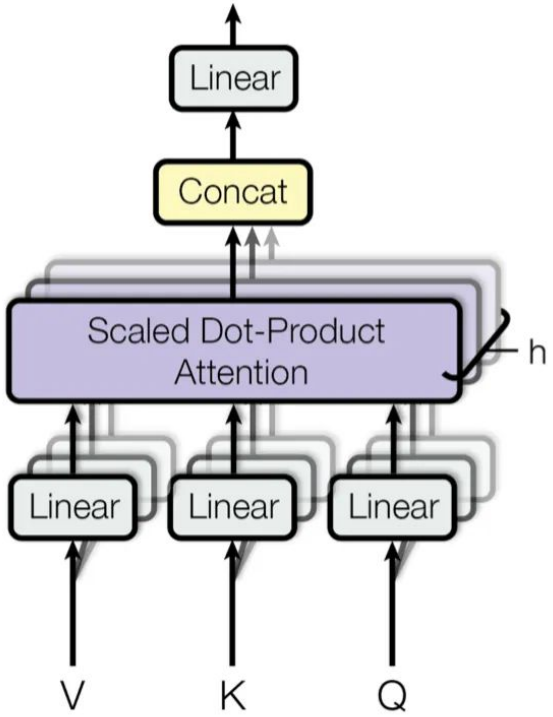
Georgia Tech

Background: Self Attention



Background: Multi-Head Attention

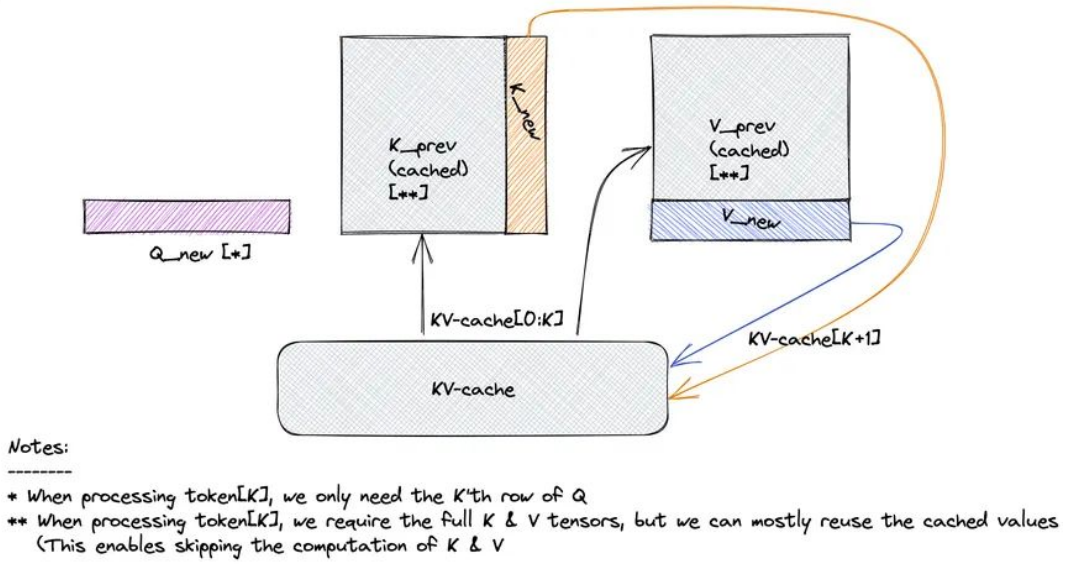


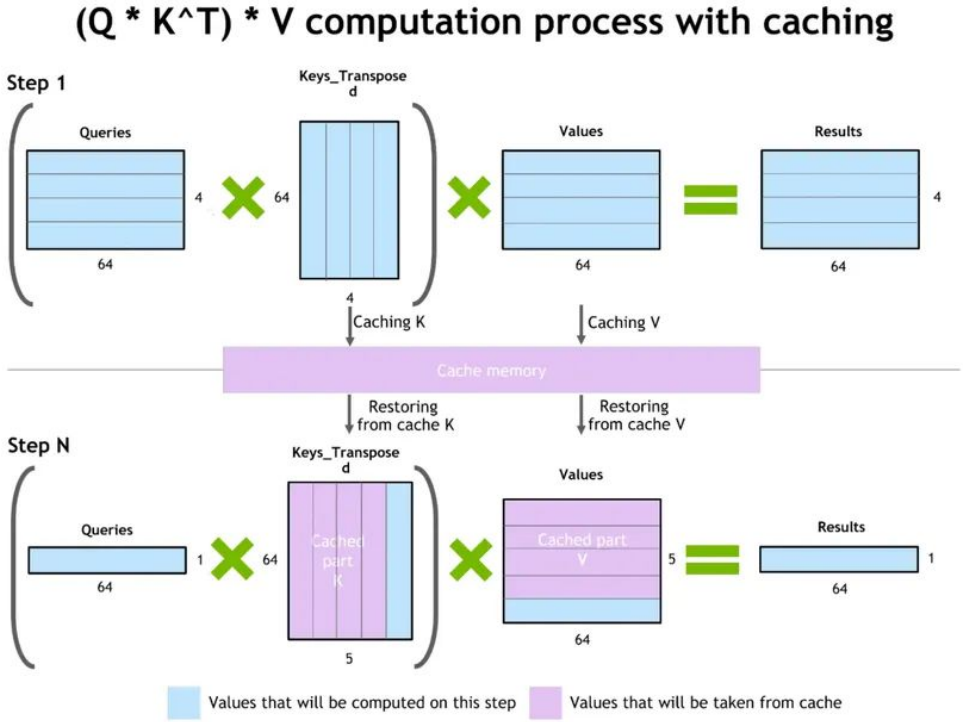


Different heads pay attention to different parts of the context vectors to learn richer representations

Georgia Tech

Background: KV caching with MHA





$$\text{Per-token KV Cache Size} = 2 \cdot L \cdot H \cdot d_h \cdot p$$

$$\text{Total KV Cache Memory} = 2 \cdot B \cdot T \cdot L \cdot H \cdot d_h \cdot p$$

Georgia Tech

What makes MHA expensive at inference?

- Each decode step reads all past K,V

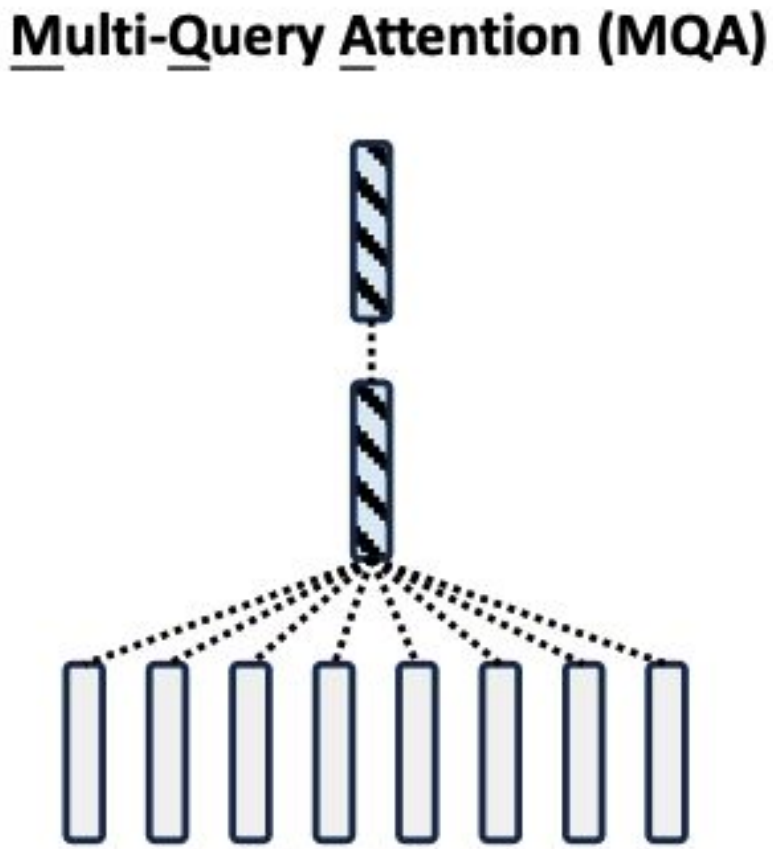
- Cost grows linearly with context, per token, per layer

$$\text{Total KV Cache Memory} = 2 \cdot B \cdot T \cdot L \cdot H \cdot d_h \cdot p$$

Multi-Query Attention

- Instead of separate Key and Value for each head, all Q heads share one averaged K and V
- All head outputs are concatenated and projected back to the output space
- Memory Advantage: huge advantage in KV cache size - stores only one set instead of h
- This is especially beneficial for long context inference where KV cache dominates memory

$$\text{head}_h = \text{Softmax}\left(\frac{Q_h K^T}{\sqrt{d}}\right)V$$



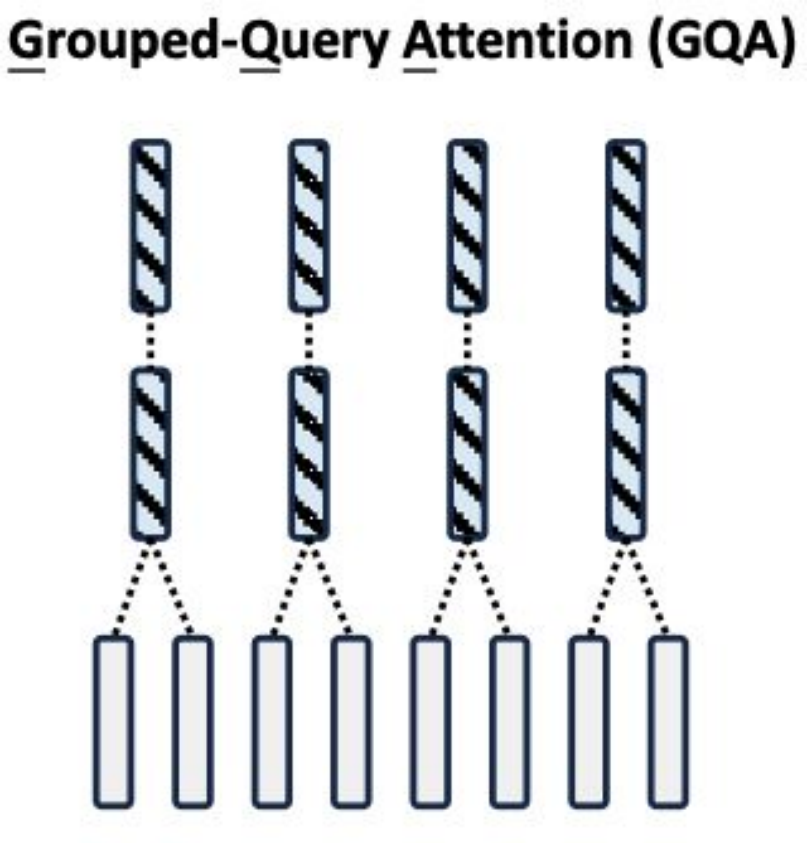
Georgia Tech

Where does MQA Succeed and Fail?

Successes	Failures
- Massive Memory Reduction by a factor of H xory with G	- Reduced Expressivity due no per-head memory
- Faster Decoding due to better cache locality	- Worse Perplexity at Scale
- Simpler Implementation: One KV Buffer	- Limited Head Specialization: only 1 head for K and V to determine different patterns

Grouped-Query Attention

- Instead of separate Key and Value for each head, groups are created for each K, V matrix
- Memory Advantage: stores G sets of K, V matrices but reduces multiplicity not dimensionality
- Fine-grained Specialization is possible through tuning G



$$\text{head}_h = \text{Softmax}\left(\frac{Q_h K_g^T}{\sqrt{d}}\right) V_g$$

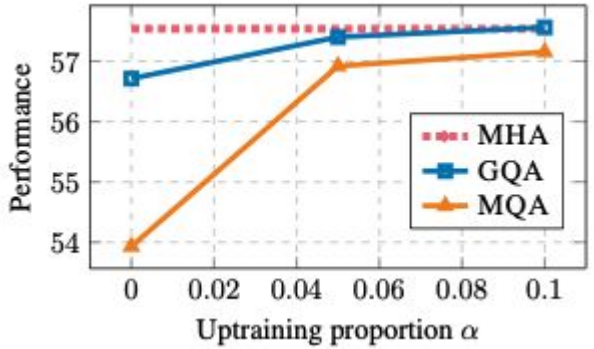
Georgia Tech

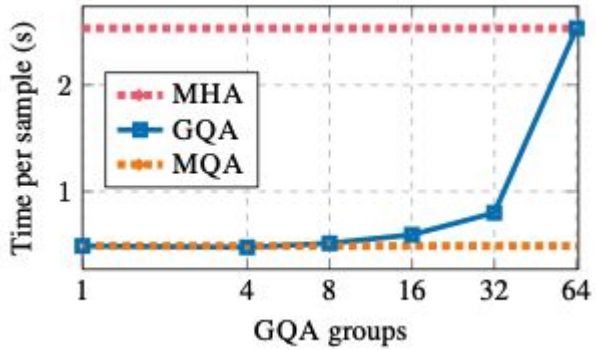
Where does GQA Succeed and Fail?

Successes	Failures
- Tuneable Trade-off: Balance Quality and Memory with G	- Storage still grows with context length; extreme scaling limited
- Better Quality than MQA by retaining head-level diversity	- No True Compression: K, V still full-dimensional
- Practical: widely adopted in LLaMA-style models	- Manual Hyperparameter: Tuned through testing

What's Still Missing?

- GQA is popular in practice because it is stable, has diversity, and is predictable; however, inference time still grows too large
- Is explicit K, V caching needed at all?
- Is K, V caching redundant?
- Can K, V be replaced with Latent Memory?



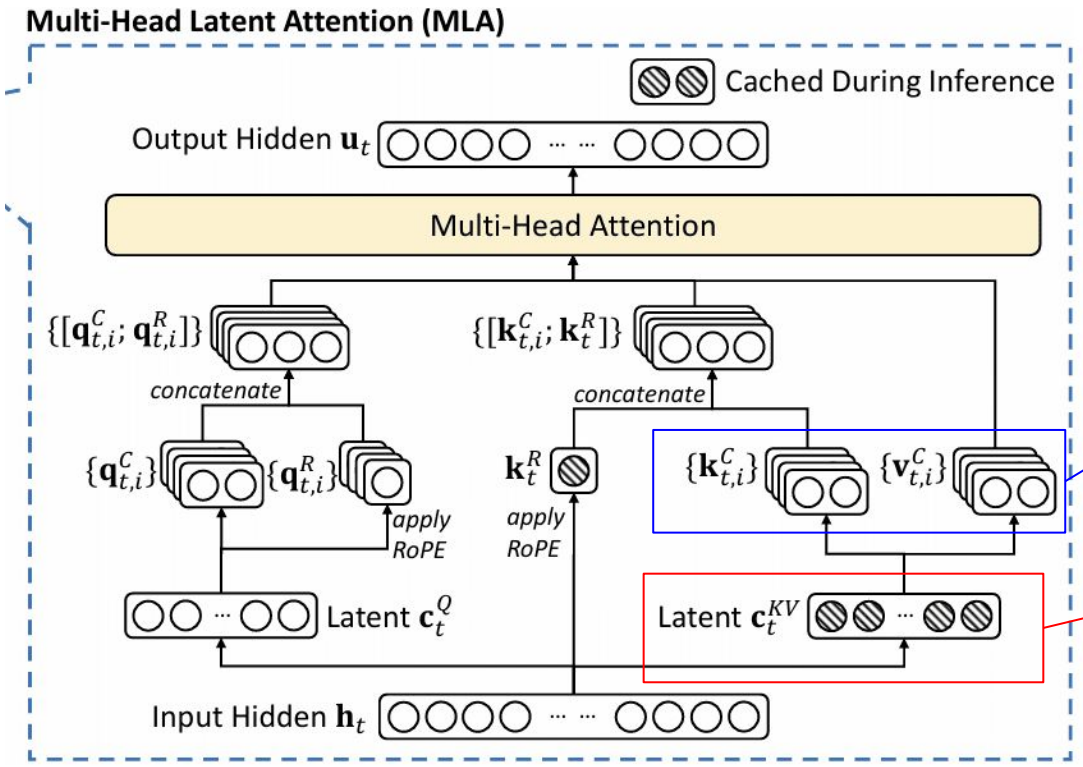


$$\mathrm{KV}_{\mathrm{MQA}} = 2 \cdot B \cdot T \cdot L \cdot d_h \qquad \mathrm{KV}_{\mathrm{GQA}} = 2 \cdot B \cdot T \cdot L \cdot G \cdot d_h$$

Georgia Tech

Multi-Head Latent Attention: Big picture

- Key realization is that you don't actually need to store full K and V.
- Have the cache store a compressed latent space representation of K and V



Reconstruct K and V matrices from cached latent representation

Compress to latent representation (cached)

Georgia Tech

So how does MLA work?

- The core of MLA is the low-rank joint compression for keys and values to reduce KV cache:

$$\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t,$$
$$\mathbf{k}_t^C = W^{UK}\mathbf{c}_t^{KV},$$
$$\mathbf{v}_t^C = W^{UV}\mathbf{c}_t^{KV},$$

$h_t \in \mathbb{R}^d$ $\quad$ $c_t^{KV} \in \mathbb{R}^{d_c}$

$W_D^{KV} \in \mathbb{R}^{d_c \times d}$ $\quad$ $d_c \ll d_h n_h$

$W_U^K, W_U^V \in \mathbb{R}^{(d_h n_h) \times d_c}$

- In order to reduce the activation memory during training, they also compress the queries:

$$\mathbf{c}_t^Q = W^{DQ}\mathbf{h}_t,$$
$$\mathbf{q}_t^C = W^{UQ}\mathbf{c}_t^Q,$$

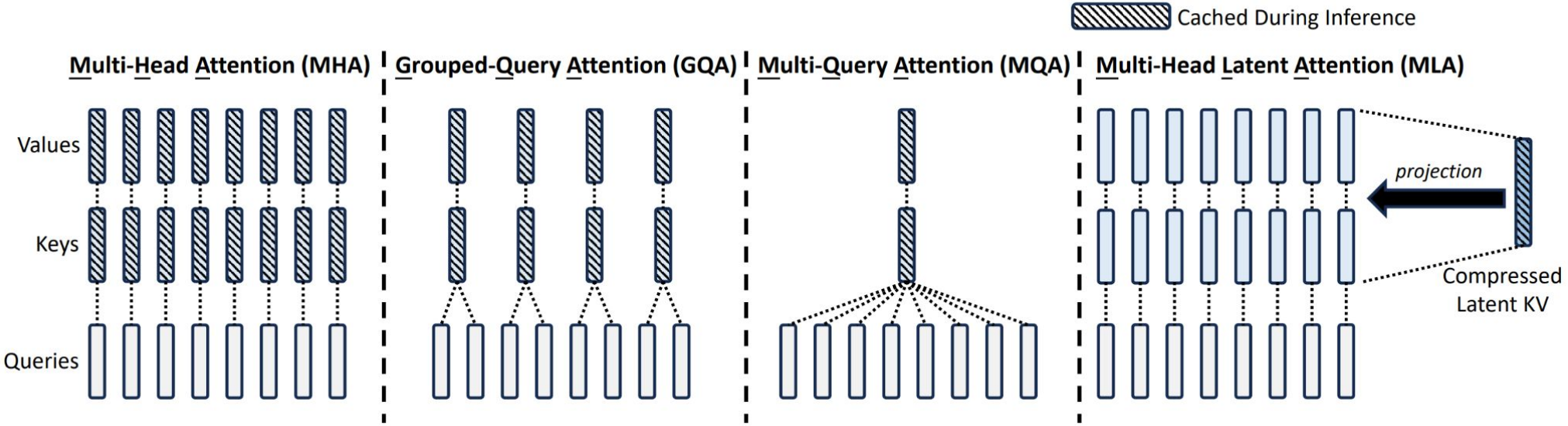
$c_t^Q \in \mathbb{R}^{d'_c}$

$W_D^Q \in \mathbb{R}^{d'_c \times d}$ $\quad$ $d'_c \ll d_h n_h$

$W_U^Q \in \mathbb{R}^{(d_h n_h) \times d'_c}$

Georgia Tech

Cache Comparison



Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2} d_h l$	Stronger

For DeepSeek-V2, d_c is set to $4d\_h$ and $d\hat{}R\_h$ is set to $d\_h/2$.

So, its KV cache is equal to GQA with only 2.25 groups, but its performance is stronger than MHA.

Georgia Tech

Experiments and Results

Benchmark (Metric)	# Shots	Dense 7B w/ MQA	Dense 7B w/ GQA (8 Groups)	Dense 7B w/ MHA
# Params	-	7.1B	6.9B	6.9B
BBH (EM)	3-shot	33.2	35.6	**37.0**
MMLU (Acc.)	5-shot	37.9	41.2	**45.2**
C-Eval (Acc.)	5-shot	30.0	37.7	**42.9**
CMMLU (Acc.)	5-shot	34.6	38.4	**43.5**

Table 8 | Comparison among 7B dense models with MHA, GQA, and MQA, respectively. MHA demonstrates significant advantages over GQA and MQA on hard benchmarks.

Ablation of MHA, GQA, and MQA

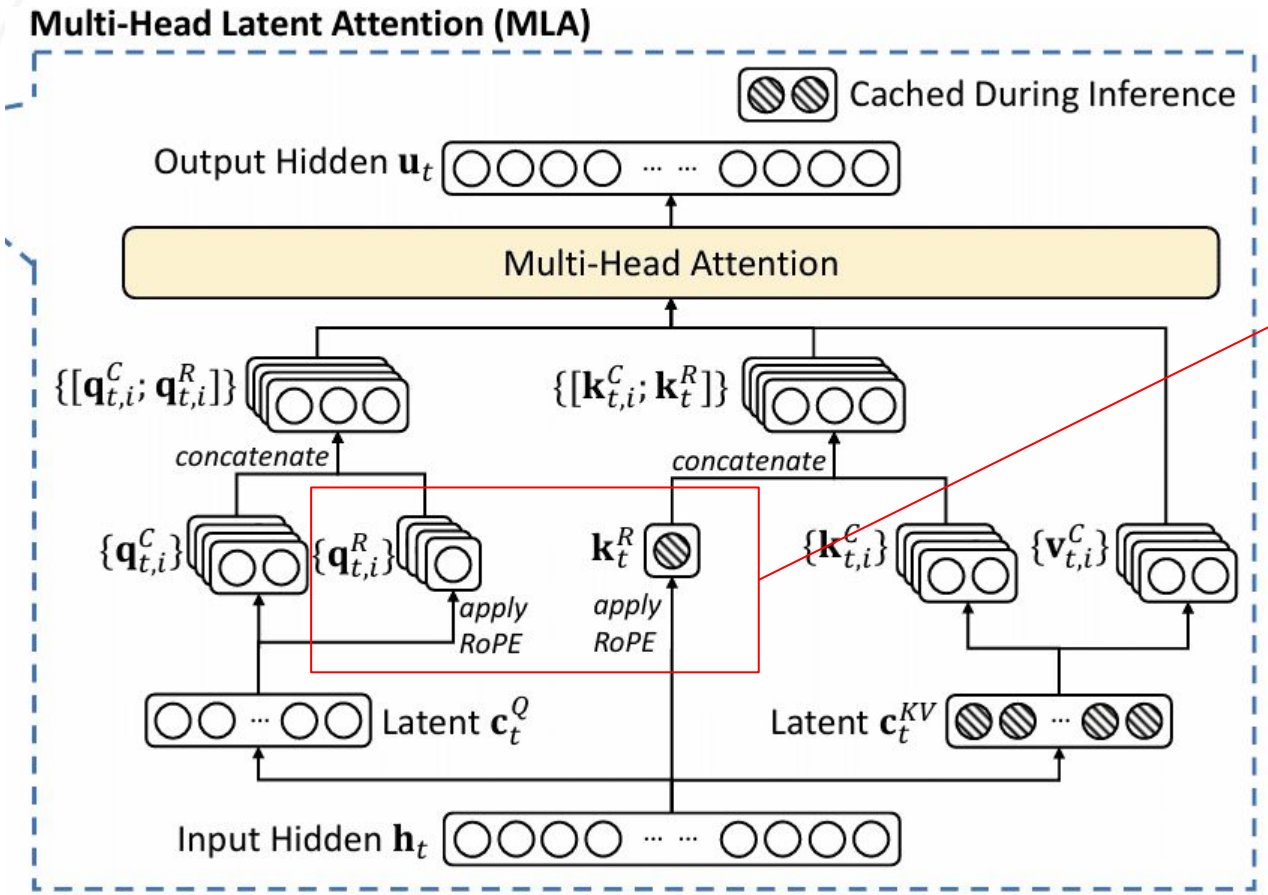
Benchmark (Metric)	# Shots	Small MoE w/ MHA	Small MoE w/ MLA	Large MoE w/ MHA	Large MoE w/ MLA
# Activated Params	-	2.5B	2.4B	25.0B	21.5B
# Total Params	-	15.8B	15.7B	250.8B	247.4B
KV Cache per Token (# Element)	-	110.6K	15.6K	860.2K	34.6K
BBH (EM)	3-shot	37.9	**39.0**	46.6	**50.7**
MMLU (Acc.)	5-shot	48.7	**50.0**	57.5	**59.0**
C-Eval (Acc.)	5-shot	**51.6**	50.9	57.9	**59.2**
CMMLU (Acc.)	5-shot	52.3	**53.4**	60.7	**62.5**

Table 9 | Comparison between MLA and MHA on hard benchmarks. DeepSeek-V2 shows better performance than MHA, but requires a significantly smaller amount of KV cache.

MHA vs MLA

Georgia Tech

RoPE: Rotary Positional Encoding



What's going on here?

RoPE: Rotary Positional Encoding

- RoPE explains how positional information is encoded geometrically in standard attention

$$Q_h = XW_Q^h, \quad K_h = XW_K^h$$

$$R_{2i}(p) = \begin{pmatrix} \cos(\theta_i p) & -\sin(\theta_i p) \\ \sin(\theta_i p) & \cos(\theta_i p) \end{pmatrix}$$

$$\widetilde{Q}_h = R(p)Q_h, \quad \widetilde{K}_h = R(p)K_h$$

$$\mathrm{RoPE}(x, p) = R(p)x$$

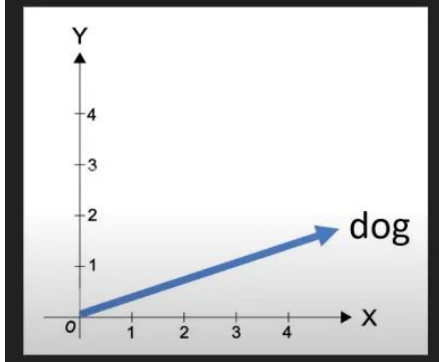


Fig 1.
(2,D) Position encoding
at position 1

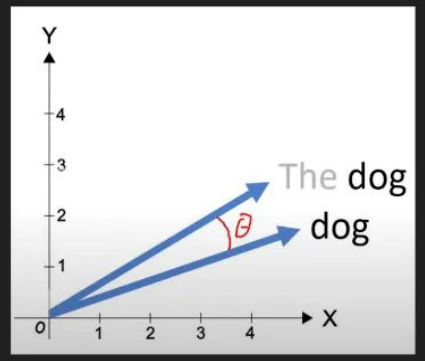


Fig 2.
(2,D) Position encoding
at position 2

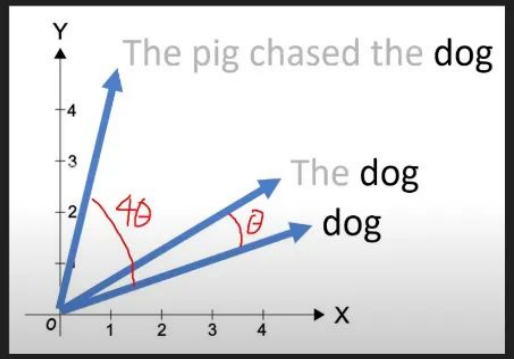


Fig 3.
(2,D) Position encoding
at position 5

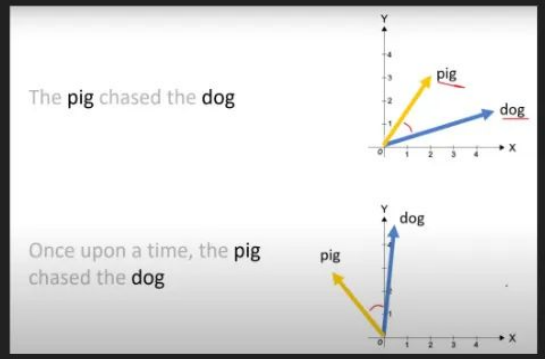


Fig 4.
(2,D) Consistent difference in positional encoding of 2 words that are two words apart, regardless of sentence length

Georgia Tech

Why RoPE fails under MLA Compression?

- MLA replaces explicit K, V with z = C(K, V)
- RoPE assumes everything is in dimension d, but the projection does not preserve inner products and destroys orthogonality
- Angles become distorted
- Relative Positions are lost

$$Q^\top R(\Delta p) K$$

$$\text{Softmax}\left(\frac{Q^\top P^\top P K}{\sqrt{d}}\right)$$

Georgia Tech

Decoupled RoPE

- Decoupled RoPE allows for the geometry to be separated from the content

$$Q = [Q_c \mid Q_p], \quad K = [K_c \mid K_p]$$

- RoPE is only applied to the positional part

$$\widehat{Q}_p = R(p)Q_p, \quad \widehat{K}_p = R(p)K_p$$

- Cache becomes significantly smaller and attention is preserved

$$(z, \widehat{K}_p) \qquad \alpha_{ij} = (Q_{c,i}^\top z_j) + (\widehat{Q}_{p,i}^\top \widehat{K}_{p,j})$$

Georgia Tech

Why does Decoupled RoPE work?

- Decoupled RoPE isolates the fragile subspace and protects it
- Positional Channel remains unchanged, so relative encoding is exact

$$\widehat{Q}_p^\top \widehat{K}_p = Q_p^\top R(\Delta p) K_p$$

- Content channel contains no position, so cache is invariant to shifts

$$z_j = C(K_{c,j})$$

- Attention remains the same just with compression on the first term

$$Q^\top K = Q_c^\top K_c + Q_p^\top K_p$$

Putting it all together

$$\mathbf{c}_t^Q = W^{DQ}\mathbf{h}_t,$$
$$[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \ldots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C = W^{UQ}\mathbf{c}_t^Q,$$
$$[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \ldots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R = \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q),$$
$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R],$$
$$\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t,$$
$$[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \ldots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C = W^{UK}\mathbf{c}_t^{KV},$$
$$\mathbf{k}_t^R = \mathrm{RoPE}(W^{KR}\mathbf{h}_t),$$
$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R],$$
$$[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \ldots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C = W^{UV}\mathbf{c}_t^{KV},$$
$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C,$$
$$\mathbf{u}_t = W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \ldots; \mathbf{o}_{t,n_h}],$$

Decoupled RoPE

Cached

Georgia Tech

Critical Perspective

- Inference gains come at the cost of added training complexity and compute
- Model quality may be sensitive to the chosen KV compression rank
- RoPE: Separating content and positional subspaces complicates the attention design
- Benefits are tightly coupled to autoregressive decoding and KV caching

Main takeaways

- KV cache is the main inference bottleneck
- MLA compresses K and V to reduce KV vache size significantly without losing information
- Presents special version for RoPE (decoupled RoPE) to work with MLA
- Results show that MLA + decoupled RoPE outperforms MHA while obtaining significant space savings

Future Work

- Non-linear projection spaces for richer semantics
- Combine MLA with different types of context windows
- Adaptive and Learned Latent Compression to improve quality even more