ICML 2026

Do Transformers Need Three Projections?

A systematic study of QKV projection-sharing variants

> **Finding.** Tying Key and Value (Q-K=V) halves the inference KV cache for +3.1% perplexity at 300M (+2.48% at 1.2B), and is orthogonal to GQA/MQA — stacking to 96.9% cache reduction for on-device / edge inference.

Ali Kayyam · **Anusha Madan Gopal** · **M. Anthony Lewis**

BrainChip Inc., Laguna Hills, CA, USA

MOTIVATION

Do we need three separate projections?

Standard self-attention learns three projections per token — Query, Key, Value. CNNs, RNNs and SSMs use more unified representations. We test three hard weight-tying constraints, holding **everything else fixed**, and ask what each projection contributes.

Baseline (QKV): $A = \mathrm{softmax}(\alpha \cdot QK^T)\, V$

Q=K-V

tie Query & Key

$A = \mathrm{softmax}(\alpha \cdot KK^T)\, V$

Symmetric attention map (KK^T). No cache benefit.

Q-K=V

tie Key & Value

$A = \mathrm{softmax}(\alpha \cdot QK^T)\, K$

Asymmetric map preserved. Caches K only → 50% cache.

Q=K=V

single projection

$A = \mathrm{softmax}(\alpha \cdot KK^T)\, K$

Most constrained: symmetric map + bottleneck.

$(X)^+$ variants add a fixed 2D sinusoidal positional encoding to restore directional asymmetry — applied only to non-causal tasks (vision, synthetic), since causal masking already enforces asymmetry in language modeling.

METHOD

One change only: which projections are tied

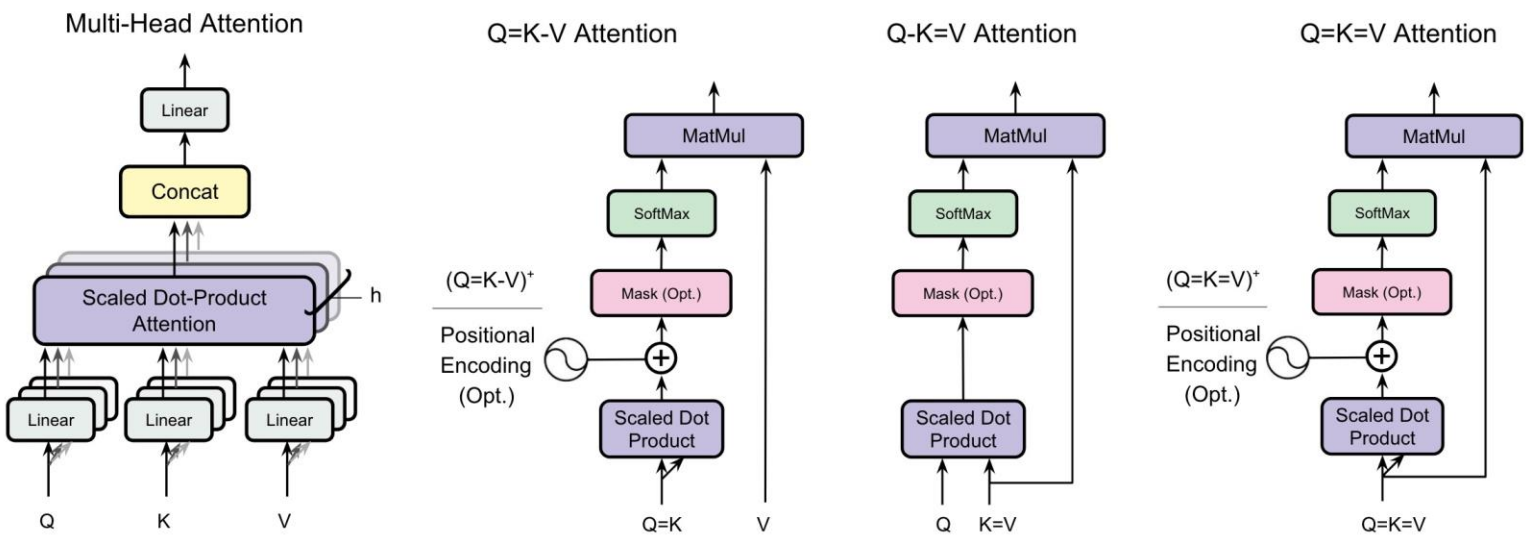


Figure 1 — attention block under each constraint; the surrounding transformer (depth, width, MLP, norm) is identical across variants.

Projection cost vs. QKV

Variant	Compute	Params
QKV	$3nd^2$	$3d^2$
Q=K-V / Q-K=V	$2nd^2$	$2d^2$
Q=K=V	nd^2	d^2

Projection ops only; the shared $O(n^2d)$ score computation is excluded.

Self-attention projections are ≈30% of parameters — so the decisive gain is inference-time KV cache, not parameter count.

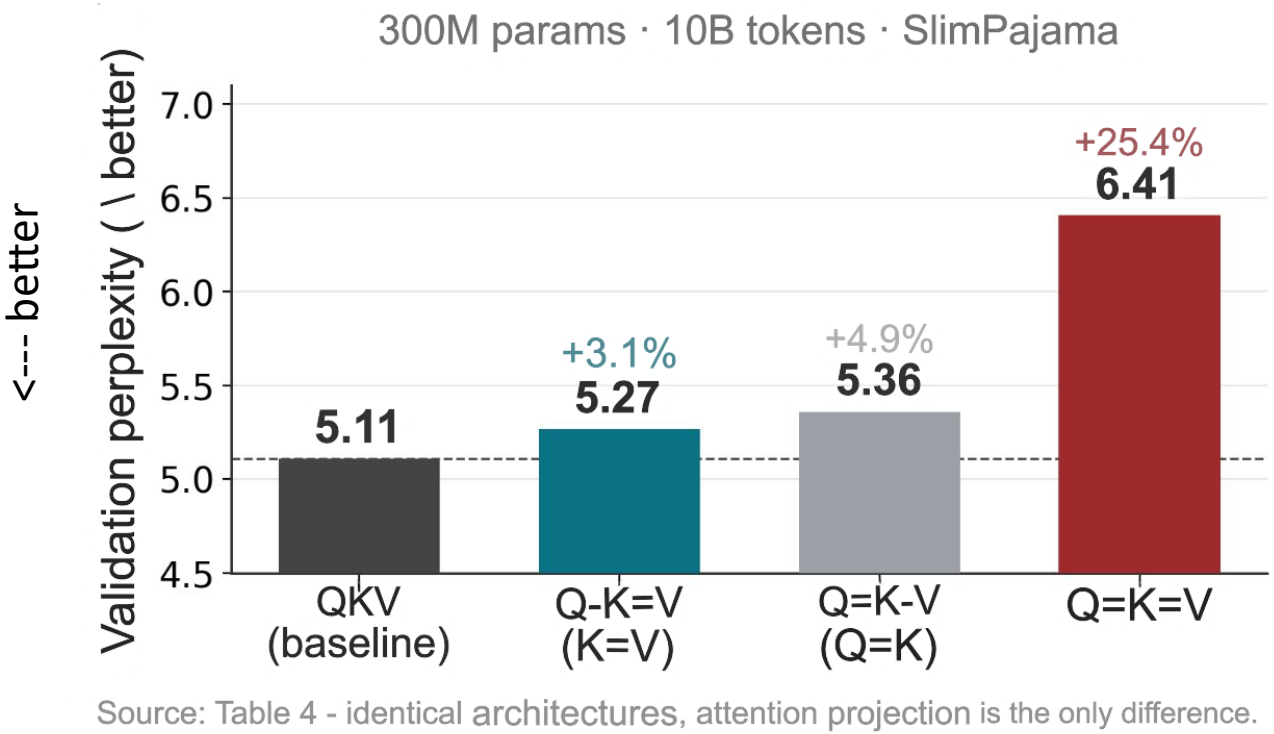
EXPERIMENTAL SETUP

A controlled, single-variable comparison

Only the attention projection differs across variants — architecture, data, optimizer, and schedule are held identical, so any difference is attributable to the projection constraint.

Task	GPT-style autoregressive language modeling (next-token prediction). Also evaluated on 5 synthetic reasoning tasks and 6 vision tasks (12 tasks, 3 domains).
Data	SlimPajama (deduplicated RedPajama), ~10B training tokens; held-out 10M-token subset for validation perplexity.
Model — 300M	20 layers, d = 1024, 16 heads (head dim 64), FFN 4096.
Model — 1.2B	22 layers, d = 2048, 32 heads (head dim 64), FFN 8192.
Common	Vocab 50,304 (GPT-2 tokenizer), tied input/output embeddings, learned absolute positions, max seq len 2048, pre-norm LayerNorm (ε = 1e-5), GELU, residual dropout 0.1.
Optimizer	AdamW (β_1 = 0.9, β_2 = 0.95, weight decay 0.1, grad clip 1.0). LR: 1000-step warmup → 6e-5, cosine decay → 6e-6.
Hardware	8 × NVIDIA A100 40GB, bfloat16, DDP, gradient accumulation 36. Steps: 4,238 (300M) / 8,475 (1.2B), ~10B tokens each.
Evaluation	Validation PPL every 500 steps. Downstream: EleutherAI lm-eval-harness, 5-shot — ARC-C, ARC-E, HellaSwag, PIQA, WinoGrande.

Q-K=V is the best projection-sharing variant



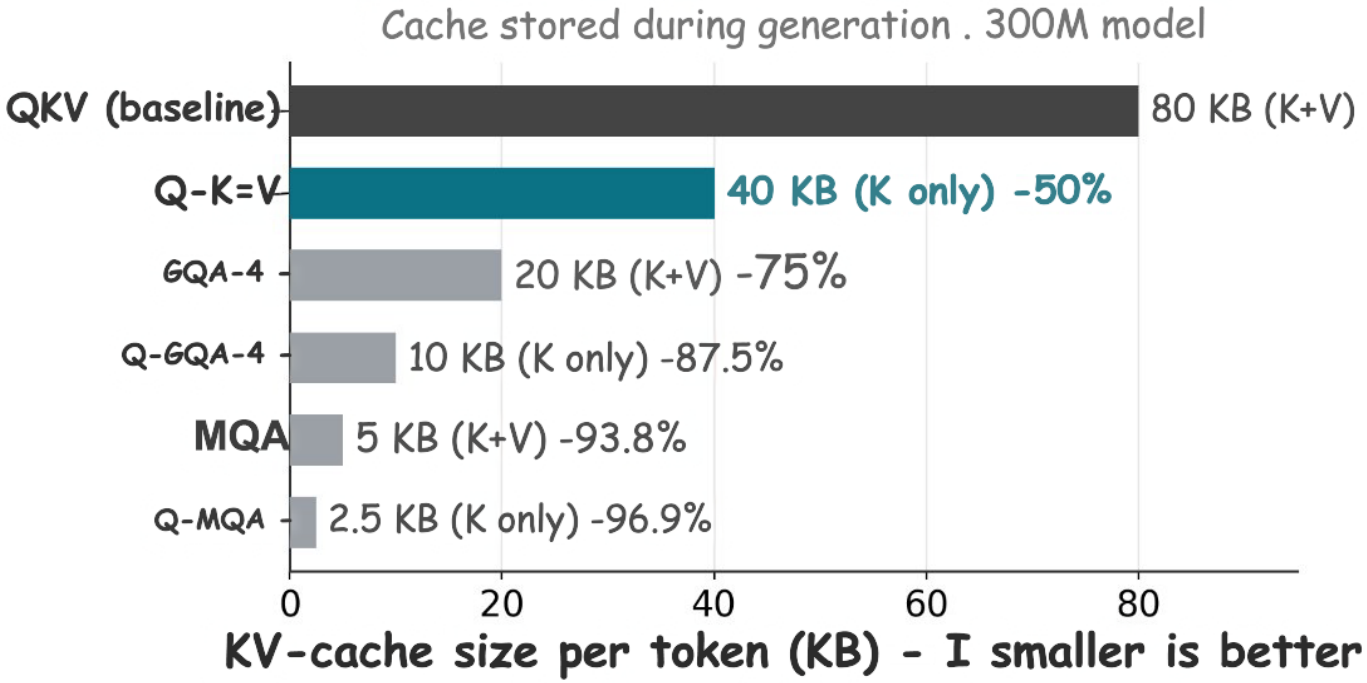
Source: Table 4 - identical architectures, attention projection is the only difference.

Variant	Val PPL	Δ vs QKV	Cache↓
QKV	5.11	—	0%
Q-K=V	**5.27**	**+3.1%**	**50%**
Q=K-V	5.36	+4.9%	0%
Q=K=V	6.41	+25.4%	50%

The Value projection is least critical. At identical params and FLOPs, tying K=V (+3.1%) costs less than tying Q=K (+4.9%) — and only K=V removes a cached tensor.

Q=K-V trains well but stores both K and V → no deployment benefit. Q=K=V collapses.

Cache size is the real benefit — and it stacks



Source: Table 7. Reduction = vs. QKV baseline (80 KB/token). Projection sharing caches K only; head sharing cuts #KV heads.

Q-K=V caches K only → 50% cut.
Q=K-V caches both → 0% (no deployment gain, despite good training quality).

At scale (per user):
32K ctx 2.62 GB → 1.31 GB · 128K ctx 10.49 GB → 5.24 GB

Orthogonal to head sharing.
K=V combines with GQA/MQA: Q-GQA-4 → 87.5%, Q-MQA → 96.9%.

Measured (A100, 1.2B, bf16):
Q-K=V − 6.5–6.9% peak memory, + 4.4–5.3% decode throughput (Tables 14–15).

Why it matters — edge / on-device. KV cache, not parameters, is the binding memory constraint on-device.

ANALYSIS

Why K=V works and Q=K doesn't

Q-K=V preserves quality

Keys and Values can share representational space while the attention map QK^T stays asymmetric.

Evidence from trained QKV models

cos(K, V) = 0.73 — highly redundant

effective rank 687 vs 702 / 1024 — similar

Q stays distinct: cos(Q,K)=0.42, cos(Q,V)=0.31

Q=K-V breaks directionality

Forcing Q = K makes the attention map symmetric (KK^T), destroying the causal directionality language modeling needs.

And it gives nothing back

+4.9% perplexity, 0% cache benefit (still stores K and V)

Q=K=V inherits both flaws → +25.4%

Takeaway. Attention needs asymmetry between Q and a shared K–V representation — not three fully independent projections.

Holds at 1.2B; the perplexity gap is not a capability gap

Validation perplexity (Table 9)

Model	PPL	Δ %	Cache↓
QKV	5.004	—	0%
Q-K=V	5.128	+2.48	50%
GQA-8	5.030	+0.52	76%
MQA	5.057	+1.06	97%
Q-GQA-8	5.158	+3.08	88%
Q-MQA	5.212	+4.16	98.5%

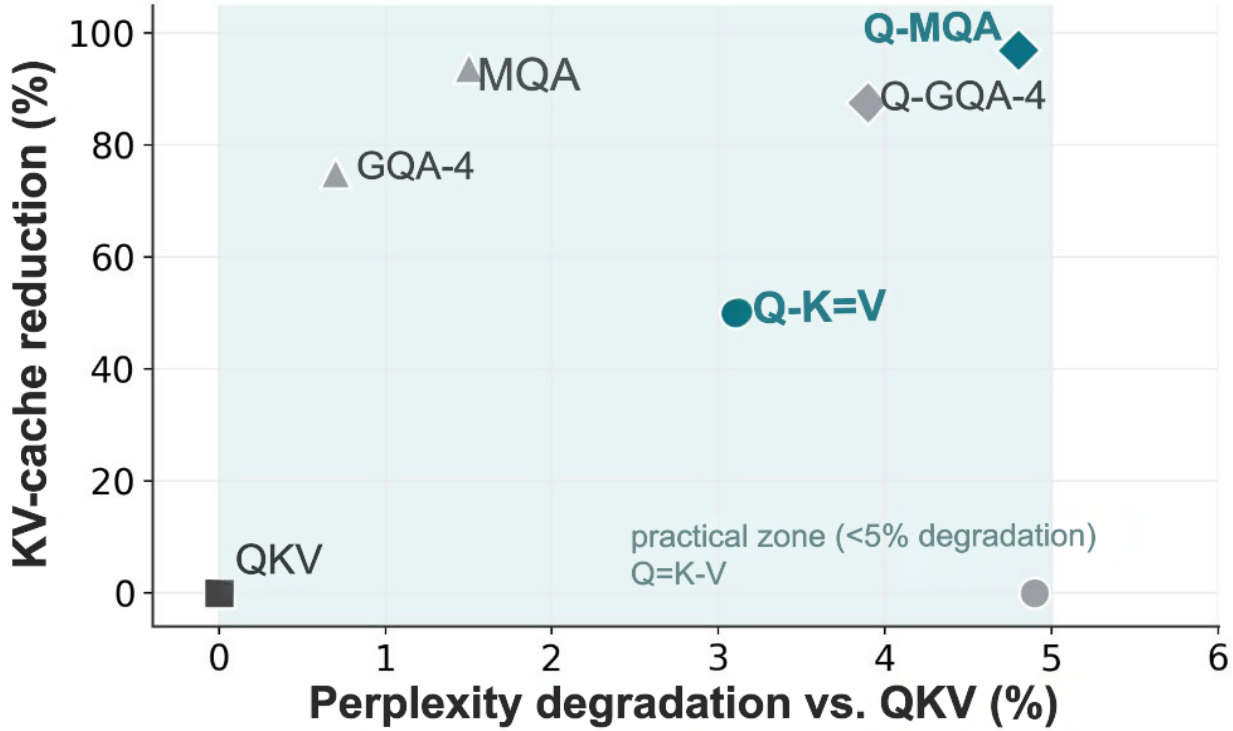
5-shot downstream accuracy (Table 11)

Model	Avg Acc (%)	Cache↓
QKV	36.40	—
Q-K=V	35.99	50%
GQA-8	35.86	75%
Q-GQA-8	36.72	87.5%
Q-MQA	34.38	96.9%

Capability: Accuracy on five standard zero-/few-shot benchmarks using the EleutherAI lm-eval-harness

Perplexity ≠ capability. Q-K=V trails QKV by 2.48% perplexity but only 0.41% downstream accuracy — while halving the cache. Relative rankings are stable from 300M to 1.2B, and Q-K=V degradation shrinks with scale (3.1% → 2.48%).

Projection and head sharing trace a clean frontier



300M results (Table 13). Q=K=V omitted (+25.4% degradation, off-scale). Q-K=V fills the gap between QKV and head sharing.

Two complementary axes.
Head sharing cuts the number of KV heads; projection sharing ties K=V. They combine multiplicatively.

Stacked operating points (300M)

Q-GQA-4 → 87.5% cache, +3.9% PPL

Q-MQA → 96.9% cache, +4.8% PPL

All variants converge smoothly under the same initialization and schedule — no training instabilities, no special tuning.

CHOOSING A VARIANT

Pick the operating point your memory budget allows

There is no single “best” variant — practitioners derive the configuration their available KV-cache memory allows. As the budget shrinks, move down the table for more compression at a bounded, measured quality cost.

KV-cache budget	Variant	Cache reduction	Δ perplexity	Why
Generous	**GQA-4**	75%	+0.7%	Smallest quality cost; quality-first deployments
Moderate	**Q-K=V**	50%	+3.1%	One constraint; stacks with the rows below
Tight	**Q-GQA-4**	87.5%	+3.9%	K=V applied within each GQA group
Very tight	**Q-MQA**	96.9%	+4.8%	Most aggressive; bounded <5% cost

All figures are measured 300M validation-perplexity degradation and KV-cache reduction (Table 13). On pure perplexity, head sharing (GQA-4) is strong on its own; the contribution of projection sharing is that it is a separate, stackable axis that extends head sharing to the most aggressive tiers.

CONCLUSION

Transformers don’t need three projections

1. **K = V is sufficient** 50% KV-cache reduction for +3.1% perplexity (300M) / +2.48% (1.2B), and only −0.41% on 5-shot downstream accuracy.
2. **Asymmetry is the requirement** Quality needs Q distinct from a shared K–V. Tying Q = K makes attention symmetric and breaks causal directionality.
3. **Orthogonal and stackable** Combines with GQA/MQA on a separate axis — up to 96.9% cache reduction at <5% perplexity cost — enabling practical edge / on-device inference.

Limitations

Largest validated scale is 1.2B (7B+ unconfirmed). The explanation for why K=V preserves quality is empirical, not formal. Evaluation reaches sequence length 2048 (no length-extrapolation study). A Q=V ablation is omitted (Q is not cached; its role differs from V).

Code & full training configs: `github.com/Brainchip-Inc/Do-Transformers-Need-3-Projections`