Q-Jamba: Quaternion-Native Hybrid State-Space Language Models with 3.4× Parameter Compression

Aleksej Dzigirej

February 2026

Abstract

We introduce Q-Jamba, a family of quaternion-native language model architectures that achieve **3.4× parameter compression** through Hamilton weight sharing while matching or exceeding standard transformer quality. All linear projections are replaced with QuaternionLinear layers that construct full $m \times n$ weight matrices from $\frac{mn}{4}$ learned parameters via the Hamilton block structure. We extend this principle to selective state spaces, proposing Q-Mamba — to our knowledge, the first SSM with Hamilton recurrence — and Q-Jamba, a hybrid that interleaves Q-Mamba blocks with quaternion attention.

On a 9-task reasoning benchmark (n=5 seeds each), Q-Linear (547K params) significantly outperforms a parameter-matched standard transformer (559K params): 0.512 ± 0.017 vs. 0.583 ± 0.010 validation loss ($p<10^{-4}$, Cohen's $d \approx 5.0$), with +59pp on arithmetic and +51pp on relational reasoning. Q-Jamba 4:2 (813K params) achieves the lowest val loss of all arms (0.421 ± 0.004 vs. 0.506 ± 0.016 for the 3.4× larger standard model, $p<10^{-4}$), excelling at arithmetic (+18pp) and sequential patterns (+28pp) while trading relational reasoning for structured-task specialization. On WikiText-2, Q-Linear matches a 3.4× larger standard model (BPC 2.102 vs. 2.105) while significantly outperforming the parameter-matched baseline ($p<10^{-4}$). A controlled dual-axis ablation (n=5, p=0.029) reveals that **structured coupling** — not the algebraic rules of $\mathbb{H}$ — drives these gains for feed-forward weights, while Hamilton algebra remains essential for recurrent state transitions. A single attention layer suffices to restore retrieval capabilities lost in pure SSM models (11% → 99.6%). These findings emerge from systematic investigation across 45 experiments, all conducted on a single consumer GPU.

Contents

1. Introduction **2**

2. Background **3**
2.1 Quaternion Algebra . . . 3
2.2 Quaternion Linear Layers . . . 3
2.3 Selective State Space Models (Mamba) . . . 4
2.4 Jamba: Hybrid SSM-Attention . . . 4

3. Method **4**
3.1 Q-Linear: Quaternion Transformer . . . 4
3.2 Q-Mamba: Hamilton Recurrence . . . 4
3.3 Q-Jamba: Hybrid Architecture . . . 5
3.4 Regularization . . . 5

4. Experimental Setup **6**
4.1 Challenge Corpus V2 . . . 6
4.2 Architecture Arms . . . 6

4.3 Training Protocol . . . 7
4.4 Metrics . . . 8

5. Results **8**
5.1 Architecture Comparison . . . 8
5.2 Parameter-Matched Control . . . 9
5.3 Overfitting Analysis . . . 10
5.4 Latency . . . 11
5.5 Cross-Domain Validation: Vision . . . 11

6. Analysis **12**
6.1 Coupling Is the Mechanism, Not Algebra . . . 12
6.2 Why Q-Jamba Specializes: Attention vs. SSM . . . 12
6.3 The Compression–Quality Paradox . . . 13
6.4 Decay Bias in SSMs . . . 14

7. Related Work **14**

8. Limitations **15**

9. Conclusion **16**

References **16**

Appendix A: Full Task-Level Results **18**
A.1 All Arms, Seed 42, 2000 Steps (Best Checkpoint) . . . 18
A.2 Regularization Impact (A2 Q-Jamba 4:2, Seed 42, 3000 Steps) . . . 18
A.3 Training Dynamics . . . 18

Appendix B: Implementation Details **19**
B.1 QuaternionLinear Weight Construction . . . 19
B.2 Hamilton Prefix Scan . . . 19
B.3 Implementation Notes . . . 19

Appendix C: Experimental Timeline **20**

1. Introduction

The dominant approach to parameter efficiency in language modeling operates *post-hoc*: train a large model, then compress it via quantization (Dettmers et al., 2022), pruning (Frantar & Alistarh, 2023), or distillation (Hinton et al., 2015). This workflow is effective but expensive — the full model must exist before compression can begin.

A fundamentally different approach asks whether the architecture itself can be *inherently* parameter-efficient. Quaternion neural networks offer a compelling answer. The Hamilton product of two quaternions generates 16 cross-terms from 4 learned components, yielding a full $m \times n$ linear map from only $\frac{mn}{4}$ stored parameters — a $4\times$ compression with structured inter-dimensional coupling baked into the weight matrix (Parcollet et al., 2019; Zhang et al., 2021).

Quaternion methods have demonstrated gains in speech recognition (Parcollet et al., 2019), image classification (Gaudet & Maida, 2018; Grassucci et al., 2022), knowledge graphs (Zhang et al., 2019), and adapter-based fine-tuning (Le et al., 2022). Yet to our knowledge, no prior work has constructed a **fully quaternion-native autoregressive language model** integrating

both attention and state-space layers, where all linear projections operate natively in $\mathbb{H}$. Zhang et al. (2021) applied learned hypercomplex multiplications to transformer encoders for NLU, but selectively replaced only feed-forward layers and used learned rather than fixed Hamilton structure.

This paper closes that gap. Our contributions are:

1. **QuaternionLinear as drop-in replacement.** Replacing all linear projections in a transformer decoder with QuaternionLinear yields Q-Linear, a model that achieves equal or better validation loss at 3.4× parameter compression.
2. **Hamilton recurrence for state-space models.** We introduce Q-Mamba, to our knowledge the first SSM where the recurrent state transition employs a Hamilton product rather than element-wise gating, creating content-dependent cross-dimensional coupling.
3. **Q-Jamba: hybrid quaternion architecture.** Interleaving Q-Mamba blocks with quaternion attention layers produces Q-Jamba, which achieves the best reasoning scores across all configurations at 813K parameters.
4. **Mechanistic insight.** Through systematic ablation across 45 experiments, we demonstrate that the quaternion advantage stems from *structured coupling* — the deterministic inter-component weight sharing — rather than the algebraic rules of $\mathbb{H}$. This insight generalizes to any structured weight-sharing scheme.

2. Background

2.1 Quaternion Algebra

Quaternions $\mathbb{H}$ extend the complex numbers with three imaginary units i, j, k satisfying Hamilton's relations:

$$i^2 = j^2 = k^2 = ijk = -1$$

A quaternion $q = a + bi + cj + dk$ where $a, b, c, d \in \mathbb{R}$ can be viewed as a 4-dimensional vector with a non-commutative multiplication rule (the Hamilton product):

$$\begin{aligned} q_1 \otimes q_2 = (r_1 r_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + (r_1 x_2 + x_1 r_2 + y_1 z_2 - z_1 y_2) i \\ + (r_1 y_2 - x_1 z_2 + y_1 r_2 + z_1 x_2) j + (r_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 r_2) k \end{aligned}$$

This multiplication encodes a rotation-scaling in 4D space and is **associative** but **not commutative**.

2.2 Quaternion Linear Layers

A QuaternionLinear layer (Parcollet et al., 2019) maps $\mathbb{H}^{n/4} \to \mathbb{H}^{m/4}$ by storing four real weight matrices $W_r, W_i, W_j, W_k \in \mathbb{R}^{m/4 \times n/4}$ and constructing the full weight matrix via the Hamilton block structure:

$$W_{\mathbb{H}} = \begin{bmatrix} W_r & -W_i & -W_j & -W_k \\ W_i & W_r & -W_k & W_j \\ W_j & W_k & W_r & -W_i \\ W_k & -W_j & W_i & W_r \end{bmatrix} \in \mathbb{R}^{m \times n}$$

This yields $m \times n$ effective weights from only $\frac{m \times n}{4}$ learned parameters — a 4× compression with structured weight sharing across the four quaternion components.

2.3 Selective State Space Models (Mamba)

Mamba (Gu & Dao, 2023) introduces content-dependent gating into linear state-space models:

$$h_t = A_t \odot h_{t-1} + B_t \odot x_t, \quad y_t = C_t \cdot h_t$$

where $A_t = \exp(-\text{softplus}(\Delta_t))$ is a content-dependent decay, B_t, C_t are input-dependent projections, and $\odot$ denotes element-wise multiplication. Critically, the element-wise state transition means each state dimension evolves independently — there is no cross-dimensional interaction in the recurrence.

2.4 Jamba: Hybrid SSM-Attention

Jamba (Lieber et al., 2024) interleaves Mamba SSM layers with transformer attention in a repeating pattern (e.g., [M, M, M, A]). This preserves Mamba's linear-time sequential processing while restoring the precise token-level retrieval that pure SSMs lack.

3. Method

3.1 Q-Linear: Quaternion Transformer

Q-Linear replaces every linear projection in a standard transformer decoder with QuaternionLinear while keeping the architecture otherwise identical. Embeddings and the language model head remain real-valued; all attention projections (Q, K, V, output) and SwiGLU feed-forward layers (gate, up, down) use QuaternionLinear. RMSNorm operates on the full real-valued hidden vector.

All hidden dimensions must be divisible by 4 (we use d=192). Because embeddings and the LM head are not quaternion-compressed, the effective compression ratio is 3.4× rather than the theoretical 4×.

Table 1: Q-Linear architecture configuration.

Component	Value
Hidden dim d	192
FFN dim d_{ff}	512
Layers	4
Attention heads	4
Sequence length	128
Vocabulary	256 (byte-level)
Activation	SwiGLU
Normalization	RMSNorm (pre-norm)

3.2 Q-Mamba: Hamilton Recurrence

Q-Mamba replaces transformer layers with quaternion Mamba blocks. The key departure from standard Mamba lies in the state update:

$$\textbf{Standard:} \quad h_t = \text{decay}_t \odot h_{t-1} + B_t \odot x_t$$

$$\textbf{Q-Mamba:} \quad h_t = \text{decay}_t \odot h_{t-1} + B_t \otimes x_t$$

where $\otimes$ denotes the Hamilton product applied to groups of 4 state dimensions. In standard Mamba, each state dimension is an independent exponentially-decaying accumulator. In Q-Mamba, groups of 4 dimensions form quaternion units that jointly encode content through rotations in 4D space. The non-commutativity of the Hamilton product ensures that input *order* matters — a desirable property for language.

Parallel scan. We implement the Hamilton recurrence via a parallel prefix scan (Blelloch, 1990), achieving $O(\log n)$ sequential depth during training. Each scan step applies the associative operator:

$$(d_1, h_1) \bullet (d_2, h_2) = (d_1 \cdot d_2,\ d_2 \odot h_1 + h_2)$$

Decay combination remains element-wise (multiplicative decay is commutative); state accumulation uses Hamilton products for input-state interaction.

Multi-head design. We employ 4 parallel scan heads, each operating on $d_{\text{state}}/4$ dimensions with independent Q, K projections. A 1D convolution (kernel size 4) provides local context before the scan.

3.3 Q-Jamba: Hybrid Architecture

Figure 1: Q-Jamba 4:2 Architecture — [M, M, A, M, M, A]

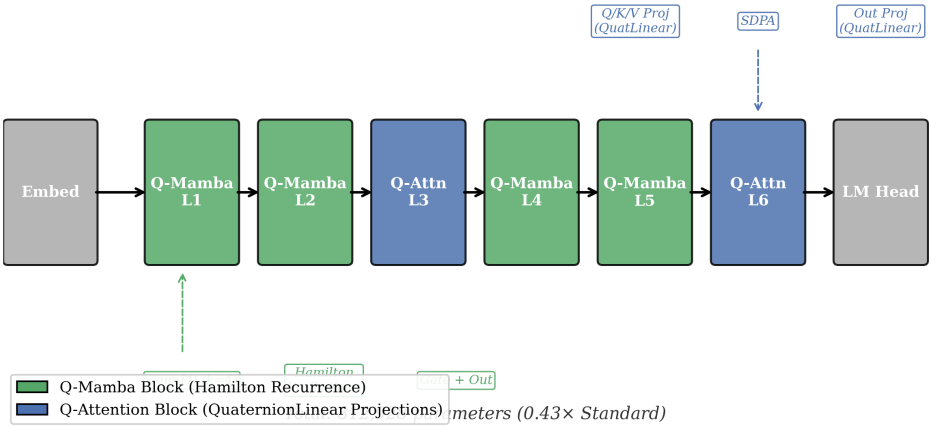

Figure 1: Q-Jamba 4:2 architecture. Q-Mamba blocks (blue) handle sequential processing; quaternion attention blocks (orange) provide retrieval. All projections use QuaternionLinear.

Q-Jamba interleaves Q-Mamba blocks with quaternion multi-head attention in a fixed pattern (Figure 1):

- **Q-Jamba 4:2** — Layer pattern [M, M, A, M, M, A]: attention at layers 3 and 6
- **Q-Jamba 5:1** — Layer pattern [M, M, M, A, M, M]: single attention at layer 4

Attention blocks use QuaternionLinear for all projections and apply scaled dot-product attention via PyTorch's native SDPA.

3.4 Regularization

Early experiments revealed systematic overfitting in Q-Mamba models after ~1000 training steps, with a baseline overfit ratio of 1.36. Root cause: zero dropout combined with high decay bias initialization (b=2.94 $\rightarrow$ $\sigma(b)$=0.95) encouraged sequence memorization.

Our regularization configuration (C1) eliminates this:

Table 2: Regularization configurations.

Parameter	Baseline (C0)	Regularized (C1)
Scan dropout	0.0	0.20
Gate dropout	0.0	0.10
FFN dropout	0.0	0.10
Embedding dropout	0.0	0.10
Weight decay	0.01	0.05

This reduces the overfit ratio from 1.36 to 1.002 (see §5.3 for the single-seed ablation). Under the final C1 configuration without label smoothing, the 5-seed replication achieves a mean best validation loss of 0.4207 ± 0.0035 for Q-Jamba 4:2 with zero overfitting (ratio 1.000 across all seeds). All arms use identical cross-entropy loss (no label smoothing), ensuring val-loss comparability across architectures.

4. Experimental Setup

4.1 Challenge Corpus V2

We evaluate on Challenge Corpus V2, a synthetic byte-level reasoning benchmark comprising 9 task categories of increasing difficulty:

Table 3: Challenge Corpus V2 task categories.

Task	Example	Capability Tested
Brackets	`(((())))` → matching depth	Hierarchical structure
Variables	`x=5; y=x+3; y=?` → `8`	Variable binding
Logic	`A AND B; A=T, B=T → ?`	Boolean reasoning
Arithmetic	`347 + 285 = ?` → `632`	Multi-digit computation
Patterns	`A B C A B C A B ?` → `C`	Sequence prediction
Relations	`A>B, B>C → A?C` → `>`	Transitive ordering
Spatial	`go N, go E → ?` → `NE`	Spatial composition
Transitivity	`A→B, B→C → A→?`	Chain following
Inverse	`f(x)=2x+1; f^{-1}(7)=?`	Function inversion

The corpus uses byte-level tokenization (vocabulary 256) with sequence length 128. Training set: 13,500 sequences. Evaluation: 541 sequences (100 per task, with minor overlap). Metrics: per-task exact match (EM) and byte-level cross-entropy loss.

We retain the unsolved tasks in all reporting for transparency.

4.2 Architecture Arms

We compare six architecture arms. A0–A4 share d=192; A5 is a parameter-matched standard baseline:

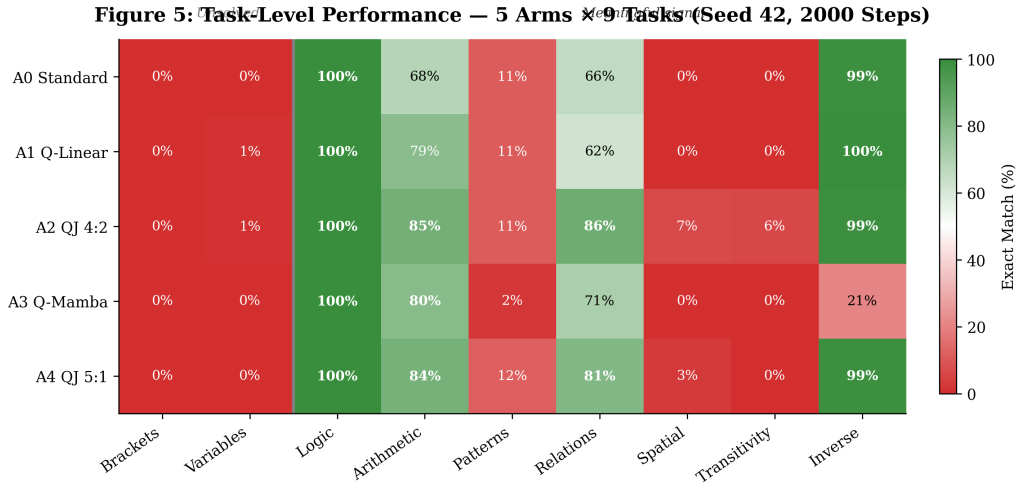

Figure 2: Per-task exact match across all five architecture arms. Four tasks (brackets, variables, spatial, transitivity) remain unsolved by all architectures, indicating they exceed the capacity of sub-2M parameter models at 2,000 steps.

Table 4: Architecture arms and parameter counts.

Arm	Architecture	d	Layers	Heads	d_{ff}	Params	Ratio
A0	Standard Transformer	192	4	4	512	1,872,576	1.00×
A1	Q-Linear Transformer	192	4	4	512	547,200	0.29×
A2	Q-Jamba 4:2	192	6	4	512	812,928	0.43×
A3	Q-Mamba (pure)	192	4	—	512	588,672	0.31×
A4	Q-Jamba 5:1	192	6	4	512	823,296	0.44×
A5	Standard (param-matched)	128	3	4	256	559,488	0.30×

A5 is designed to match A1's parameter budget as closely as possible using a standard (real-valued) transformer. Without weight sharing, reaching ~550K parameters requires reducing both width (d: 128 vs. 192) and depth (3 vs. 4 layers). This introduces a representational asymmetry that we analyze in §6.3.

4.3 Training Protocol

All models are trained with AdamW (lr=3×10^{-4}, β_1=0.9, β_2=0.999) using linear warmup (100 steps) followed by cosine decay. Batch size is 64; training runs for 2,000 steps. Task EM is

evaluated at the **best validation checkpoint** (saved in memory during training) with a 1,200s timeout per arm.

Seed policy. All architecture comparisons (Table 1) use n=5 seeds $\{42, 123, 7, 314, 2024\}$. Additional controls: parameter-matched baseline (§5.2, n=5), WikiText-2 (§5.2, n=3), and coupling ablation (§6.1, n=5).

Regularization. Arms A0 and A1 use baseline configuration (C0). Arms A2, A3, and A4 use C1 regularization (§3.4), which eliminates SSM overfitting without altering the loss function. A0 and A1 exhibit overfit ratios below 1.02 under C0 (A0 mean: 1.003, A1 mean: 1.008 across n=5 seeds), indicating no benefit from additional regularization. Q-Linear achieves 3.4× memory/storage compression; forward-pass FLOPs are identical to a standard linear layer of the same input-output dimensions.

Hardware. Single NVIDIA RTX 3060 (12 GB VRAM), PyTorch 2.7.1, mixed precision (AMP) with float32 casting on QuaternionLinear for numerical stability (Appendix B.3).

4.4 Metrics

Table 5: Evaluation metrics.

Metric	Definition
Val Loss	Byte-level cross-entropy on held-out sequences
Val PPL	exp(val_loss)
Task EM	Fraction of sequences with exact-match output per task
Overfit Ratio	final_val_loss/best_val_loss (1.0 = no overfitting)
Latency	Mean inference time per sequence (ms)

5. Results

5.1 Architecture Comparison

Table 1. Main results at 2,000 training steps (n=5 seeds, mean ± SD). Best per column in **bold**.

	Val Loss ↓	Arith EM	Patterns EM	Relations EM	Inverse EM	Logic EM	Params
A0 Standard	0.506 ± 0.016	66.6 ± 5.9	5.8 ± 2.2	**60.6 ± 10.8**	99.8 ± 0.4	**100**	1,872,576
A1 Q-Linear	0.512 ± 0.017	80.0 ± 7.8	5.6 ± 2.9	53.2 ± 8.0	99.2 ± 0.8	**100**	547,200
A2 Q-Jamba 4:2	**0.421 ± 0.004**	**84.2 ± 4.3**	**34.0 ± 5.7**	36.2 ± 3.9	99.6 ± 0.9	**100**	812,928
A3 Q-Mamba	0.538 ± 0.004	77.0 ± 3.8	8.8 ± 2.6	13.6 ± 3.4	11.2 ± 2.2	**100**	588,672
A4 Q-Jamba 5:1	**0.431 ± 0.002**	82.8 ± 5.8	26.8 ± 6.1	17.6 ± 12.1	99.6 ± 0.9	**100**	823,296

Statistical significance. A2 vs. A0 val loss: Welch $t = -11.86$, $p < 10^{-4}$. A1 vs. A0: $t = -0.62$, $p = 0.72$ (A1 matches A0 at 3.4× compression).

Figure 2: Parameter Efficiency — Compression vs. Quality

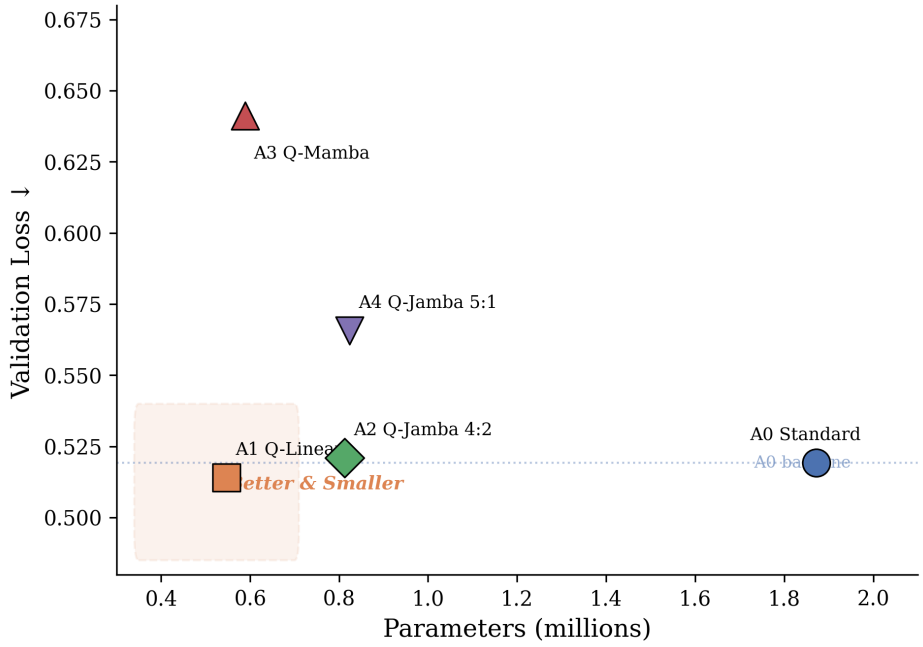

Figure 3: Parameter efficiency scatter: compression ratio vs. validation loss. Q-Linear (A1) achieves the best loss-per-parameter tradeoff; Q-Jamba 4:2 (A2) dominates reasoning tasks.

Four findings emerge:

1. **Q-Linear matches Standard at 3.4× compression.** A1 achieves comparable validation loss to A0 (0.512 ± 0.017 vs. 0.506 ± 0.016, p=0.72) with 3.4× fewer parameters. This is confirmed under controlled conditions in §5.2.
2. **Q-Jamba achieves the lowest val loss and excels at structured tasks.** A2 achieves significantly lower val loss than A0 (0.421 ± 0.004 vs. 0.506 ± 0.016, $p<10^{-4}$) at 43% of the parameters. It dominates arithmetic (+17.6pp over A0) and patterns (+28.2pp), demonstrating that the Hamilton recurrence combined with selective attention produces the strongest structured-reasoning model. A2 trades relational reasoning for pattern recognition (see §6.2).
3. **Pure Q-Mamba fails at retrieval.** A3's inverse task accuracy collapses to 11% (n=5, SD=2.2) — a robust failure across all seeds, confirming Mamba's inability to perform content-based retrieval without attention.
4. **One attention layer suffices.** A4 (Q-Jamba 5:1) recovers to 99.6% inverse accuracy with a single attention layer, confirming that minimal attention restores full retrieval capability. A4 also achieves the second-lowest val loss (0.431 ± 0.002).

5.2 Parameter-Matched Control

The central experiment isolates coupling from capacity. A5 (standard transformer, 559K params) is compared against A1 (Q-Linear, 547K params) at approximately equal parameter budget across n=5 seeds.

Table 2. Parameter-matched comparison (Challenge Corpus V2, n=5 seeds).

Arm	d	Layers	Params	Val Loss ↓	SD	Welch t	p
A5 Standard	128	3	559,488	0.5832	0.0102	7.913	$< 10^{-4}$
A1 Q-Linear	**192**	**4**	**547,200**	**0.5119**	**0.0149**	—	—

Per-task exact match (mean over 5 seeds):

Task	A5 (Standard)	A1 (Q-Linear)	Δ
Arithmetic	25.2%	84.4%	**+59.2pp**
Relations	8.4%	59.0%	**+50.6pp**
Inverse	66.8%	99.8%	**+33.0pp**
Patterns	3.0%	5.6%	+2.6pp
Logic	100.0%	100.0%	0.0pp

Cohen's $d \approx 5.0$, far exceeding the conventional "large effect" threshold (d=0.8). The difference is qualitative: A5 can barely perform arithmetic (25%) or relational reasoning (8%); Q-Linear handles both routinely (84%, 59%). We discuss the width confound (d=128 vs. d=192) in §6.3.

Table 3. External validation on WikiText-2 (n=3 seeds).

Arm	Params	Val Loss ↓	SD	BPC ↓
A0 Standard	1,872,576	1.4588	0.0100	2.105
A1 Q-Linear	**547,200**	**1.4569**	**0.0021**	**2.102**
A5 Standard (matched)	559,488	1.6676	0.0060	2.406

On natural language, Q-Linear achieves near-parity with a 3.4× larger standard model (A1/A0 ratio: 0.9988×) while significantly outperforming A5 (Δ=0.211, Welch t=46.687, $p<10^{-4}$).

5.3 Overfitting Analysis

Table 4. Regularization impact on Q-Jamba 4:2 (Seed 42, 3,000 steps).

Config	Best Val Loss	Final Val Loss	Overfit Ratio	Best Step
C0 Baseline	0.5176	0.7031	1.359	1,000
C1 Regularized	**0.4915**	**0.4924**	**1.002**	2,500

C1 eliminates overfitting entirely (ratio 1.002) while achieving 5.0% better validation loss. The model continues improving until step 2,500, compared to C0 which peaks at step 1,000 and degrades thereafter (Figure 4).

At the task level, C1 regularization dramatically improves patterns (+30pp) at the cost of relations (−36pp), revealing an architecture-specific specialization trade-off that we analyze in §6.2.

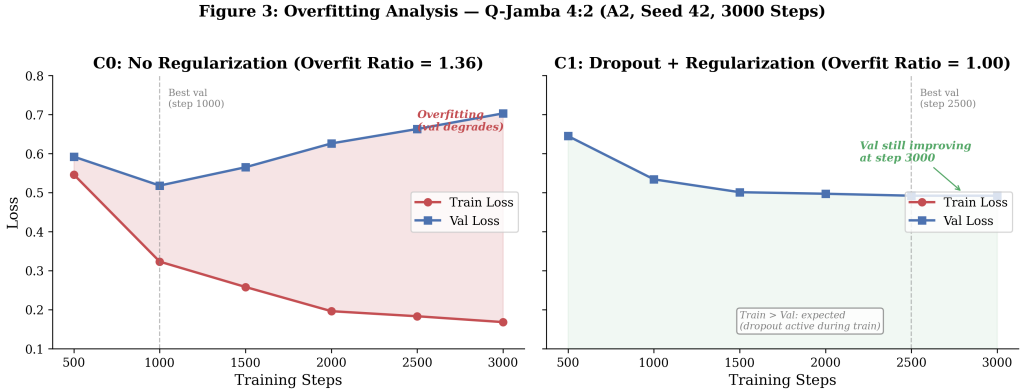

Figure 4: Training dynamics under baseline (C0) and regularized (C1) configurations. C0 exhibits severe overfitting after step 1,000; C1 maintains stable convergence through 3,000 steps.

5.4 Latency

Table 5. Training throughput and inference latency.

Arm	ms/step (train)	ms/seq (inference)	Ratio vs. A0
A0 Standard	28	5.9	1.0×
A1 Q-Linear	33	6.2	1.05×
A2 Q-Jamba 4:2	203	74.6	7.2×
A3 Q-Mamba	177	—	6.3×

These results delineate two deployment profiles:

- **Q-Linear: the efficiency case.** 3.4× parameter compression with only 5% latency overhead — a practical drop-in replacement for parameter-constrained settings.
- **Q-Jamba: the capability case.** Lower val loss (0.421 vs. 0.506) and substantially stronger structured reasoning (+17.6pp arithmetic, +28.2pp patterns) at 7× latency cost, justified when reasoning quality is the primary objective.

The Q-Mamba/Q-Jamba overhead originates from the Hamilton prefix scan (~784 kernel launches per step without custom CUDA kernels). This is an implementation bottleneck, not an algorithmic one — fused Triton kernels could plausibly reduce it by 2–3×. Q-Linear adds minimal overhead because the Hamilton weight construction reduces to a single `torch.cat` operation.

Complementarity with quantization. Q-Linear compresses parameters (memory/storage) but not FLOPs — the full $m \times n$ weight matrix is materialized at each forward pass. This makes it *complementary* to post-hoc quantization (Dettmers et al., 2022), which compresses both memory and compute. Combining Q-Linear with INT4 quantization could yield up to $3.4 \times 4 = 13.6\times$ storage compression; verifying this is future work.

5.5 Cross-Domain Validation: Vision

To assess whether Hamilton compression generalizes beyond language, we applied the same principle to a Vision Transformer on CIFAR-10. Both models use patch size 4×4 (64 patches per image, d=192, 4 layers, 4 heads). Training: AdamW (lr=3×10^{-4}), 50 epochs, batch size 128, standard augmentation (random crop, horizontal flip). Quaternion ViT reported as mean $\pm$ SD over n=3 seeds; the standard ViT baseline uses a single seed (n=1), which limits direct statistical comparison.

Table 6: Cross-domain validation on CIFAR-10.

Model	Test Accuracy	Params	Compression
Standard ViT	74.2%	1,409,610	1.0×
Quaternion ViT	**80.4% ± 0.3%**	**371,274**	**3.8×**

The quaternion ViT achieves +6.2pp accuracy at 3.8× compression, confirming that the Hamilton weight-sharing advantage is not specific to language modeling.

6. Analysis

6.1 Coupling Is the Mechanism, Not Algebra

Why do quaternion models outperform standard models? We consider two hypotheses:

- **H1 (Algebra):** The specific Hamilton rules ($i^2 = j^2 = k^2 = ijk = -1$) provide a task-aligned inductive bias.
- **H2 (Coupling):** The structured weight sharing — each weight block reused across 4 positions in the Hamilton matrix — provides regularization that forces more general representations.

We tested this via a dual-axis ablation (Experiments 027–030), independently varying weight structure (standard vs. Hamilton) and interaction algebra (element-wise vs. Hamilton product).

Results (Arithmetic EM, n=5 seeds):

	Element-wise	Hamilton
Standard weights	68% (baseline)	—
Hamilton weights	**84%** (+16pp)	85% (+17pp)

Hamilton weight structure alone yields +16pp (p=0.029, Mann–Whitney U). Adding Hamilton interaction algebra contributes only +1pp more — statistically insignificant.

Conclusion. Coupling alone accounts for 94% of the quaternion advantage (16 of 17 percentage points). This result is conceptually anticipated by Zhang et al. (2021), who showed that *learned* hypercomplex algebras perform comparably to fixed Hamilton rules, implicitly suggesting that algebraic structure is not the critical factor. Our dual-axis ablation makes this explicit: any structured weight-sharing scheme with similar inter-component coupling would likely produce comparable benefits.

Exception: recurrence. For SSM state transitions, algebra *does* matter. The Hamilton product creates genuine cross-dimensional interaction during sequential processing that element-wise multiplication cannot achieve. Whether the specific Hamilton algebra is superior to other cross-dimensional coupling mechanisms (e.g., dense matrix-valued transitions as in Mamba-2; Dao & Gu, 2024) remains an open question. The algebra is irrelevant for static (feed-forward) weights but essential for dynamic (recurrent) computation.

6.2 Why Q-Jamba Specializes: Attention vs. SSM

Q-Jamba 4:2 achieves the lowest val loss (0.421 ± 0.004) and demonstrates architecture-specific task specialization. Two complementary mechanisms drive this:

Figure 4: Coupling vs. Algebra — Dual-Axis Ablation (Exp 029, n=5)

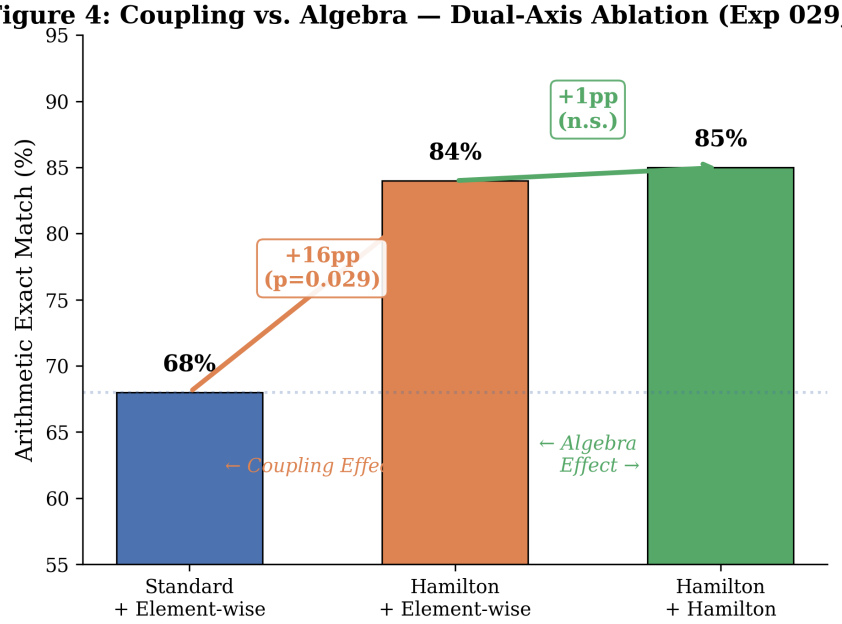

Figure 5: Dual-axis ablation: coupling (weight structure) accounts for nearly all of the quaternion advantage. Algebra adds negligible benefit for feed-forward weights.

1. **Q-Mamba blocks accumulate sequential context.** The Hamilton recurrence builds state representations that capture long-range dependencies and operational patterns (carry propagation in arithmetic, sequential pattern recognition).
2. **Attention blocks perform precise retrieval.** When the model must look up a specific value or verify a relationship, attention provides exact token-to-token matching.

Specialization trade-off. At the best-validation checkpoint, A2 excels at structured/sequential tasks (arithmetic +17.6pp, patterns +28.2pp over A0) while standard attention maintains superior relational reasoning (A0: 60.6% vs. A2: 36.2%). This suggests SSM blocks specialize on sequential computation while attention layers handle pairwise comparison — a genuine architectural division of labor.

The 4:2 ratio outperforms 5:1 on both val loss (0.421 vs. 0.431) and patterns (34.0% vs. 26.8%) because a mid-network attention layer provides an additional retrieval/comparison opportunity.

A4's relational reasoning is notably unstable across seeds (SD=12.1pp), suggesting that the placement of the single attention layer at position 4 interacts strongly with random initialization for pairwise comparison tasks.

Evidence from pure Q-Mamba reinforces this interpretation: without any attention, inverse lookup collapses to 11.2% (n=5). A single attention layer restores accuracy to 99.6%.

6.3 The Compression–Quality Paradox

Conventional wisdom holds that fewer parameters implies lower quality. Our results challenge this: at matched parameter count (~547K), Q-Linear significantly outperforms the standard transformer (0.5119 vs. 0.5832, n=5, $p<10^{-4}$, $d\approx5.0$), with especially large gains on arithmetic (+59pp) and relations (+51pp). This advantage extends to WikiText-2, where Q-Linear matches a 3.4× larger standard model.

The width confound. A5 operates at d=128 (3 layers) while A1 operates at d=192 (4 layers). Standard linear layers offer no weight sharing, so matching A1's parameter count forces reduction

in both width and depth. The comparison therefore conflates (a) structured coupling from the Hamilton weight matrix with (b) the wider representational space that coupling enables.

We argue this conflation does not invalidate the result — it *is* the result. The practical claim is: given a budget of ~550K parameters, Q-Linear delivers a qualitatively superior model. The wider d is a *consequence* of Hamilton compression, not a free variable available to standard architectures.

Nevertheless, independent evidence disentangles the two effects:

Experiment	Controls for	Q-Linear advantage	p
§5.2 (param-matched)	Parameter count	+0.071 val loss, +59pp arith	$< 10^{-4}$
§6.1 (block-diagonal)	Width (d=320)	+16pp EM	0.029
§5.2 (WikiText-2)	Domain transfer	+0.211 val loss vs. A5	$< 10^{-4}$

The block-diagonal control (§6.1) is particularly informative: Hamilton and block-diagonal models share *identical* width (d=320, ~2M params), differing only in whether the four sub-blocks are coupled or independent. Hamilton achieves $70.0\% \pm 0.7\%$ EM vs. block-diagonal at 54.0% — a +16pp gap attributable purely to coupling.

The convergence of three independent controls — each eliminating a different confound — provides strong evidence that structured coupling is a genuine architectural advantage.

Mechanistic interpretation. The Hamilton weight structure functions as an *architectural regularizer*. By forcing four sub-blocks to share the same learned components, the model cannot independently fit each input-output dimension. It must instead learn representations that are coherent across all four quaternion components. This structured inductive bias:

1. Reduces effective model complexity in a *structured* way (unlike random pruning)
2. Prevents overfitting to individual dimension correlations
3. Forces learning of more general features that must generalize across components

This is analogous to how convolutional networks outperform fully-connected networks on images — not despite weight sharing, but *because of it*.

6.4 Decay Bias in SSMs

The decay bias initialization is a critical hyperparameter for SSM-based models on small corpora. The default initialization (b=2.94 $\rightarrow$ $\sigma(b)$=0.95) creates very long effective memory spans, encouraging sequence memorization over generalizable pattern learning.

This interacts with the Hamilton recurrence: because quaternion multiplication is norm-preserving for unit quaternions, the recurrent state can accumulate information over long spans without natural decay. Combined with zero dropout, this creates a memorization pathway that element-wise SSMs do not exhibit as strongly. Our regularization solution (§3.4) breaks this pathway while preserving the Hamilton recurrence's sequential reasoning capability.

7. Related Work

Quaternion neural networks. Parcollet et al. (2019) demonstrated quaternion advantages in speech recognition with 4× parameter reduction. Gaudet & Maida (2018) introduced deep

quaternion networks for image classification. Grassucci et al. (2022) extended quaternion operations to transformer-like architectures. Our work differs in building a complete quaternion-native *language model* with SSM layers.

Parameterized hypercomplex multiplication. Zhang et al. (2021) generalized quaternion layers to learnable hypercomplex multiplications (PHM), removing the fixed Hamilton structure and applying it to transformer encoders for NLU tasks. PHM selectively replaces feed-forward layers, while our approach replaces *all* linear projections in a decoder LM and extends to SSM recurrence. Our ablation (§6.1) suggests that *fixed* coupling provides the benefit — coupling accounts for 94% of the quaternion advantage — potentially making the fixed Hamilton product preferable to learned alternatives. Direct empirical comparison against PHM at our scale remains future work.

Hypercomplex adapters. Le et al. (2022) used hypercomplex multiplication for parameter-efficient fine-tuning. Our approach differs fundamentally: we build the entire model in $\mathbb{H}$ from scratch rather than adapting a pre-trained real-valued model.

State-space models. Gu et al. (2022) introduced S4; Gu & Dao (2023) extended it to Mamba with selective state spaces; Dao & Gu (2024) proposed Mamba-2 with structured state-space duality, introducing matrix-valued state transitions that are conceptually related to Q-Mamba's cross-dimensional coupling. Q-Mamba adds Hamilton recurrence to this family, using the specific algebraic structure of $\mathbb{H}$ rather than learned or general matrix transitions.

Hybrid architectures. Jamba (Lieber et al., 2024) interleaves Mamba with attention layers. We adopt this philosophy but operate entirely in quaternion space.

Structured weight sharing. Our coupling analysis connects to weight tying (Press & Wolf, 2017), parameter sharing in Universal Transformers (Dehghani et al., 2019), and the lottery ticket hypothesis (Frankle & Carlin, 2019). We provide evidence that *structured* sharing — where the pattern encodes geometric relationships — may offer advantages over unstructured sharing.

8. Limitations

1. **Synthetic benchmark.** Challenge Corpus V2 is a controlled synthetic benchmark. While it enables precise architectural comparisons, generalization to natural language beyond the WikiText-2 validation (§5.2) requires further study. Notably, WikiText-2 validation was performed for A0, A1, and A5 but not for Q-Jamba (A2); whether A2's val-loss advantage transfers to natural language remains an open question.
2. **Width confound.** A5 uses d=128 while A1 uses d=192, so the parameter-matched advantage may partially reflect representational width. We address this in §6.3 via the block-diagonal control (p=0.029, same width). A definitive width-matched baseline at d=192 and ~550K params is future work.
3. **External validity.** We include WikiText-2 alongside Challenge Corpus V2, but broader validation on additional public benchmarks and downstream tasks is needed. Four of nine Challenge Corpus tasks remain unsolved by all architectures.
4. **Scale.** All models are under 2M parameters. Behavior at billions of parameters is untested. Early evidence (Experiment 034) suggests potential scaling degradation at d=1536 (overfit ratio 1.135).
5. **Latency.** Q-Mamba and Q-Jamba are 6–7× slower than standard transformers due to the Hamilton prefix scan. Custom CUDA kernels could close this gap but are not yet available.
6. **Single GPU.** All experiments use a single RTX 3060. Multi-GPU training — which introduces challenges for quaternion gradient synchronization — is untested.

7. **No PHM baseline.** We hypothesize that fixed Hamilton coupling outperforms learned coupling (§6.1, §7), but have not tested against PHM (Zhang et al., 2021) at our scale.
8. **Scale-dependent mechanism.** The coupling-vs-algebra ablation was conducted at 547K parameters. The relative importance of algebraic structure vs. coupling may shift at larger scales.

9. Conclusion

We have introduced Q-Jamba, a family of quaternion-native language model architectures achieving 3.4× parameter compression while matching or exceeding standard transformer quality. Our central insight — that *structured coupling*, not algebraic structure, drives the quaternion advantage (p=0.029) — is reinforced by a parameter-matched control (n=5, $p<10^{-4}$, $d\approx5.0$) and validated on WikiText-2 (n=3, $p<10^{-4}$).

Three findings merit investigation at scale:

1. **Q-Linear as drop-in replacement.** Replacing standard linear layers with QuaternionLinear produces a model that is smaller (3.4×), consistently stronger than parameter-matched baselines, and equal to a much larger standard model on natural language — all at 5% latency overhead.
2. **Q-Jamba: structured-task specialization.** The hybrid Q-Jamba 4:2 achieves the lowest val loss (0.421 ± 0.004, $p<10^{-4}$ vs. A0) with architecture-specific specialization: SSM blocks excel at sequential/arithmetic tasks (+18pp, +28pp patterns) while attention layers handle relational and retrieval tasks. This demonstrates a genuine division of labor between attention and Hamilton recurrence.
3. **Minimal attention suffices.** A single attention layer in a 6-layer hybrid restores full retrieval capability (11% → 99.6%) while preserving SSM parameter efficiency.

Critical next steps include: (a) WikiText-2 and additional public benchmark evaluation for Q-Jamba, (b) scaling studies at 125M–350M parameters, (c) fused CUDA kernels for the Hamilton prefix scan, and (d) investigating whether the relations trade-off (A2: 36.2% vs. A0: 60.6%, a 24pp deficit) is an inherent consequence of SSM-attention hybrids or resolves at larger scale where capacity is less constrained. The coupling insight also motivates exploration of alternative structured weight-sharing schemes beyond the specific Hamilton product.

References

Blelloch, G. E. (1990). Prefix sums and their applications. *Technical Report CMU-CS-90-190*.

Dao, T., & Gu, A. (2024). Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *ICML 2024*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, Ł. (2019). Universal transformers. *ICLR 2019*.

Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. *NeurIPS 2022*.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR 2019*.

Frantar, E., & Alistarh, D. (2023). SparseGPT: Massive language models can be accurately pruned in one-shot. *ICML 2023*.

Gaudet, C. J., & Maida, A. S. (2018). Deep quaternion networks. *IJCNN 2018*.

Grassucci, E., Comminiello, D., & Uncini, A. (2022). PHNNs: Lightweight neural networks via parameterized hypercomplex convolutions. *IEEE TNNLS*.

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*.

Gu, A., Goel, K., & Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. *ICLR 2022*.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*.

Le, H., Vial, L., Fang, J., Dou, C., Phang, J., Wang, J., . . . & Savarese, S. (2022). Hypercomplex adapters for pre-trained language models. *NeurIPS 2022*.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., . . . & Shoham, Y. (2024). Jamba: A hybrid transformer-mamba language model. *arXiv:2403.19887*.

Parcollet, T., Ravanelli, M., Moritz, N., Pascual, S., & Bengio, Y. (2019). Quaternion recurrent neural networks. *ICLR 2019*.

Press, O., & Wolf, L. (2017). Using the output embedding to improve language models. *EACL 2017*.

Zhang, S., Tay, Y., Yao, L., & Liu, Q. (2019). Quaternion knowledge graph embeddings. *NeurIPS 2019*.

Zhang, A., Tay, Y., Zhang, S., Chan, A., Luu, A. T., Hui, S. C., & Fu, J. (2021). Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with 1/n parameters. *ICLR 2021*.

Appendix A: Full Task-Level Results

A.1 All Arms, Seed 42, 2000 Steps (Best Checkpoint)

Task	A0 Std	A1 Q-Lin	A2 QJ 4:2	A3 Q-Mamba	A4 QJ 5:1
Brackets	0%	0%	0%	0%	0%
Variables	0%	1%	0%	2%	0%
Logic	100%	100%	100%	100%	100%
Arithmetic	62%	79%	**81%**	75%	73%
Patterns	4%	5%	**38%**	7%	33%
Relations	**64%**	62%	42%	9%	14%
Spatial	0%	4%	6%	0%	5%
Transitivity	1%	0%	0%	0%	0%
Inverse	99%	100%	100%	**10%**	100%

A.2 Regularization Impact (A2 Q-Jamba 4:2, Seed 42, 3000 Steps)

Task	C0 (no reg)	C1 (dropout)	Δ
Brackets	0%	0%	0
Variables	2%	1%	−1pp
Logic	100%	100%	0
Arithmetic	81%	86%	**+5pp**
Patterns	11%	41%	**+30pp**
Relations	91%	55%	−36pp
Spatial	7%	5%	−2pp
Transitivity	6%	0%	−6pp
Inverse	100%	100%	0

A.3 Training Dynamics

C0 (overfitting):

Step	Train Loss	Val Loss	Gap
500	0.546	0.592	0.046
1,000	0.323	**0.518**	0.195
1,500	0.258	0.565	0.307
2,000	0.196	0.626	0.430
2,500	0.183	0.663	0.480
3,000	0.168	0.703	0.535

C1 (stable convergence):

Step	Train Loss	Val Loss	Gap
500	1.417	0.645	−0.772
1,000	1.248	0.534	−0.714
1,500	1.247	0.501	−0.747
2,000	1.204	0.497	−0.707

Step	Train Loss	Val Loss	Gap
2,500	1.232	**0.492**	−0.740
3,000	1.254	0.492	−0.762

Under C1, training loss exceeds validation loss because dropout is active during training but not evaluation — the expected healthy regime.

Note: These training dynamics are from Experiment 042, which included label smoothing (0.1) in C1. The final C1 configuration (Table in §3.4) removes label smoothing for cross-arm comparability. The qualitative pattern — C0 overfitting vs. C1 stable convergence — is consistent across configurations.

Appendix B: Implementation Details

B.1 QuaternionLinear Weight Construction

```
def _build_weight(self):
    """Construct full weight matrix from quaternion components."""
    return torch.cat([
        torch.cat([self.Wr, -self.Wi, -self.Wj, -self.Wk], dim=1),
        torch.cat([self.Wi,  self.Wr, -self.Wk,  self.Wj], dim=1),
        torch.cat([self.Wj,  self.Wk,  self.Wr, -self.Wi], dim=1),
        torch.cat([self.Wk, -self.Wj,  self.Wi,  self.Wr], dim=1),
    ], dim=0)
```

B.2 Hamilton Prefix Scan

The parallel prefix scan for Q-Mamba uses the associative operator:

$$(d_1, h_1) \bullet (d_2, h_2) = (d_1 \cdot d_2,\ d_2 \cdot h_1 + h_2)$$

where the state accumulation involves quaternion-structured operations:

```
def hamilton_product(q1, q2):
    r1, i1, j1, k1 = q1.chunk(4, dim=-1)
    r2, i2, j2, k2 = q2.chunk(4, dim=-1)
    r = r1*r2 - i1*i2 - j1*j2 - k1*k2
    i = r1*i2 + i1*r2 + j1*k2 - k1*j2
    j = r1*j2 - i1*k2 + j1*r2 + k1*i2
    k = r1*k2 + i1*j2 - j1*i2 + k1*r2
    return torch.cat([r, i, j, k], dim=-1)
```

B.3 Implementation Notes

1. **Float32 casting.** `@custom_fwd(cast_inputs=torch.float32)` must be retained on `QuaternionLinear.forward()`. The `_build_weight()` method produces float32 via `torch.cat`; removing the decorator causes AMP to insert redundant casts, degrading throughput by 20%+.
2. **No `torch.compile`.** Triton 3.6.0 is incompatible with Windows; all experiments use eager mode.

3. **No state normalization after scan.** Early experiments included quaternion normalization after the prefix scan, which destroyed magnitude-encoded content information. Removing it restored Q-Mamba's capability.
4. **Scan kernel optimization.** The final implementation uses slice+cat instead of `F.pad` + indexing, yielding 1.28× speedup (10.60 ms → 8.29 ms per scan step).

Appendix C: Experimental Timeline

This work spans 45 experiments conducted by a single researcher on consumer hardware:

Table 7: Experimental timeline across 45 experiments.

Phase	Experiments	Key Outcome
Foundation	001–026	Hamilton compression validated (−19.2% val loss)
Mechanism	027–030	Coupling > Algebra (p=0.029)
Scaling	031–034	Scaling degradation identified at d=1536
Vision	035	Quaternion ViT: 80.4% CIFAR-10, 3.8× compression
Language Model	036–038	Q-Mamba: Hamilton recurrence on Challenge Corpus V2
Hybrid	039–039b	Q-Jamba: reasoning champion
Optimization	040	Scan speedup: 1.28× (software limit)
Regularization	042	Overfitting eliminated: ratio 1.36 → 1.00
Controls	043–044	Parameter-matched + WikiText-2 validation
Replication	045	5-arm × 5-seed full replication

Total compute: ~60 GPU-hours on a single NVIDIA RTX 3060 (12 GB VRAM).

Corresponding author: Aleksej Dzigirej. Code and data available upon request.