TOKAFORMER: ROUTING CO-LOCATED MODALITIES AS TYPED TOKENS BEFORE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal models often fuse the inputs that share a position by embedding each with one shared encoder and adding the results. This is cheap, but when track identity enters only through additive offsets it erases the pairing between each value and the input it came from, and tasks that must report what one particular input said cannot recover it. We asked when this loss matters and how to avoid it at reasonable cost. For such a shared additive encoding we prove that every reassignment of values among inputs yields the same fused vector, so no downstream model can retrieve a named input's value better than a histogram bound we compute exactly. A code that gives each input its own subspace escapes the theorem, which locates the problem in the encoder rather than in addition. Tokaformer, the architecture we study, keeps each position's inputs as separate typed tokens through a linear-recurrent routing stage and only then discards the extra tokens. On position-, content-, and track-addressed synthetic tasks, factored input succeeded where summed input failed, and the failure was not explained by sequence length or shared tuning. On aggregation tasks such as counting, neither fusion held a robust advantage, which suggests pooling only the inputs an aggregate can absorb; this hybrid reached similar observed composite accuracy to full factoring (about 0.996) with 35% fewer routing tokens. On WeatherBench 2 and BigEarthNet the ranking of the two fusions depended on the input encoder: with one encoder shared across channels factored fusion won clearly, while with a separate encoder per channel, which binds identity to value before the sum, summed fusion was as good or better, consistent with the proposed mechanism. We summarize the results as bind before you compress: keep inputs separate until the network has selected among them, or until the encoding itself has recorded which is which, and pool wherever selection is unnecessary.

1 INTRODUCTION

Compression is a decision about what to forget. When several inputs describe the same position, a model that embeds them with one shared encoder and adds them together keeps a pooled summary of their values but gives up the record of which input supplied which value. If the task is to take a maximum over sensors, that pairing is noise and can be discarded early. If the task is to report what one particular sensor measured, the pairing is the answer, and no downstream capacity can recover it once it is gone. Early additive fusion makes this choice at the input, usually without anyone deciding to make it.

We think the choice deserves to be made explicitly. Multimodal architectures already offer many ways to keep inputs apart or to pick among them, from separate modality streams to cross-attention, multiplicative fusion, and learned latent bottlenecks (Baltrušaitis et al., 2019; Tsai et al., 2019; Zadeh et al., 2017; Nagrani et al., 2021; Jaegle et al., 2021). By modality we mean any input type that shares a position with the others, be it a sensor, a spectral band, or a pressure level. Summing per-track embeddings at each position is also common practice in large models (Hayes et al., 2025). What is less clear is *when* keeping inputs separate changes what a model can learn, *where* in the network the separation is used, and *how much* of the input needs to stay separate at all. Benchmark scores alone cannot answer these questions, because in real data the factors that matter, such as how much selection the task needs and how crowded the inputs are, move together. We build task families in which each can be varied on its own, and we then check the conclusions on two real datasets.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We study how the information a fusion step preserves interacts with selective prediction, and we implement one way of delaying compression, **Tokaformer**, which keeps co-located inputs as separate typed tokens for as long as the model needs to route among them and then compresses (Fig. 1b). Each primary position carries typed *modifier* slots. A linear-recurrent stage traverses the full expanded stream, a **DROP** step discards the modifier slots, and the rest of the network runs on the primary stream. Section 2 proves what a shared additive encoding loses and shows, with a constructive control, that the loss comes from the encoding rather than from addition (Fig. 1a,c). Section 3 uses controlled synthetic tasks (Table 1, Fig. 1d–g) to separate a structural loss of binding from a difficulty of training, and locates the useful routing before DROP (Fig. 2). Section 4 shows where separate tokens stop helping, motivating a **hybrid fusion** that pools only what can be pooled, and shows that needing a single item does not make pooling safe (Fig. 3). Section 5 asks the same question of WeatherBench 2 and BigEarthNet, where which fusion wins depends on whether the input encoder already keeps track identity, consistent with the mechanism of Section 2 (Fig. 4).

2 Additive fusion and binding

2.1 Shared-sum ambiguity

To see exactly what summation loses, consider the simplest case. A position carries K named tracks (for example, K sensors), and each track holds a value v_t drawn from an alphabet of A possible values. A common way to fuse them is to embed every value with one shared encoder e, add a learned offset a_t that marks the track, and sum the results:

$$F(v) = b + \sum_{t=1}^{K} \big[e(v_t) + a_t\big],$$

where b collects everything that does not depend on the values, such as the position encoding. Throughout, we hold fixed which tracks are present and everything else the decoder sees, including the query.

Theorem 1 (shared-sum binding ambiguity). For every permutation π of the values among the fixed tracks, $F(\pi v) = F(v)$. Hence, whenever two assignments that both occur with positive probability call for different answers, no decoder that sees only F and the fixed side information can be right on both. This covers selective retrieval $y = v_J$ whenever unequal values can be swapped while the cue J stays fixed, and it holds for randomized decoders.

The proof follows from the fact that addition does not care about order. Swapping two values between tracks only reorders the terms $e(v_t)$ in the sum, and the offsets a_t and b are added regardless of which value sits where, so both assignments produce the same fused vector. Any model downstream of the fusion, however wide or deep, therefore receives identical inputs in the two cases and cannot tell them apart.

The theorem also tells us how well such a model can possibly do. If the values are drawn independently and uniformly, and the cue J is chosen independently of them, the sum depends only on how often each value occurs (the histogram H), not on where it occurs. No decoder can then do better than guess the most frequent value, so accuracy is at most $\mathbb{E}[\max_a H_a/K]$, with equality when the fused vector identifies the histogram. With $A = 16$, this ceiling is 0.531, 0.337, and 0.258 for $K = 2, 4, 8$. In our implementation, swapping values between tracks changes the fused vector by at most 5.7×10^{-6} in float32 (Appendix A).

Importantly, the theorem places the problem in the encoder rather than in addition itself. What fails is an encoder that cannot tell which track it is encoding, so that a value loses its attachment to the slot it came from. This is the binding problem (Greff et al., 2020) in its simplest form. Typed tokens avoid the ambiguity because each token carries its track identity, and so does any additive code that ties value to type, as tensor-product and holographic representations have long done (Smolensky, 1990; Plate, 1995; Kleyko et al., 2022). Finally, preserving a distinction at the input does not guarantee that a model *learns* to use it. Identifiability and learnability are separate questions (Wagstaff et al., 2019), and we treat them separately throughout.

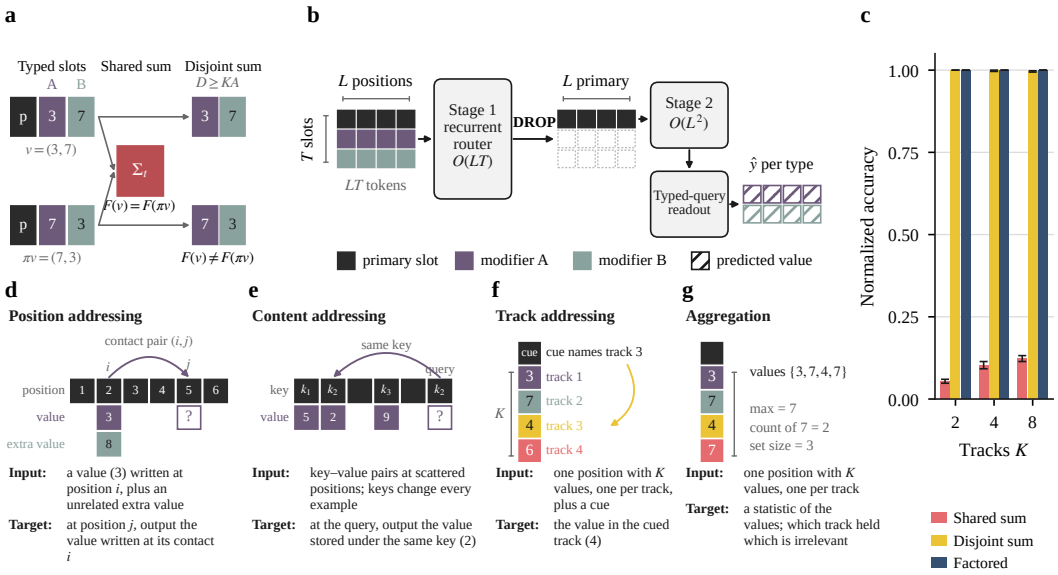


Figure 1: Binding before compression. (a) Two value assignments to tracks A and B: distinct as typed slots, identical under a shared additive encoding ($F(v) = F(\pi v)$, Theorem 1), distinct again under a disjoint code that gives each track its own coordinates ($D \geq KA$). (b) Tokaformer routes the $L \times T$ typed stream in a bidirectional recurrent stage 1 ($O(LT)$). DROP keeps the primary slot so stage 2 ($O(L^2)$) and the readout run on L tokens; hatched boxes are the predicted value per position and type. (c) Disjoint-code control at $K = 2, 4, 8$ (mean $\pm$ s.d., $n = 10$ per cell). (d–g) The synthetic tasks of Table 1: black boxes are position tokens, colored boxes modifier tokens carrying a value, the outlined box the value to output, and arrows the pairing to recover.

2.2 Architecture and the disjoint additive code

Tokaformer is built around this observation (Fig. 1b). For a sequence of L primary positions, each with T typed slots, it lays out all LT tokens position by position, so that each position occupies a contiguous block: the primary token first, followed by its $T - 1$ modifiers. Each modifier token is the sum of a value embedding, a learned type embedding, and a learned slot embedding, so its track identity travels with it. Which slot serves as the primary is set by the task (e.g., Z500 for weather, the red band for land cover, and the position token for the synthetic tasks). The network then runs in three steps. First, a bidirectional linear-recurrent mixer reads the full expanded stream, using one forward and one backward pass with shared weights whose outputs are summed. Second, DROP keeps only the primary token of each block (a stride-T gather), reducing the sequence back to L tokens. Third, stage 2 and a typed-query readout operate on these L tokens: each position gets one query per type and slot, placed at that position by rotary encoding, that cross-attends to stage-2 outputs within a one-position window. The real-data models instead read the stage-2 primaries through a linear head. The synthetic experiments use dense attention in stage 2, while the weather model is recurrent in both stages.

For the recurrent stage we use delta-rule linear attention. Rather than attending over every past token, these mixers keep a fixed-size memory matrix that is updated at each token by writing a key–value pair and read out with a query, with a gate that lets each update overwrite old content as well as add new content (Katharopoulos et al., 2020; Schlag et al., 2021; Yang et al., 2024; 2025; Kimi Team, 2025; Hatamizadeh et al., 2026). We use two existing variants, GDN2 and KDA, running on the Flash Linear Attention kernels (fla-org contributors, 2026); neither is a contribution of this work.

The reason for this ordering is cost. A linear-recurrent pass over the expanded stream scales as $O(LT)$ with sequence length (Gu & Dao, 2024), and dense attention over the L primaries scales as $O(L^2)$, whereas dense attention over the full expanded stream would scale as $O(L^2T^2)$. DROP therefore confines the expanded stream to the one stage that can afford it. It also imposes a division of labor on the model: anything the readout will need from a modifier must be carried into the re-

tained primary stream (not necessarily its own position's primary) during stage 1, since the modifiers are gone afterwards.

The theorem also suggests a second way out, which we tested directly (Fig. 1a,c). If each track's values are encoded in their own orthogonal subspace, the sum still records which track held which value, much like writing each sensor's reading in its own column of a table rather than adding all readings into one cell. With task, data, model, and parameter count matched, this disjoint code reaches 0.996 normalized accuracy at $K = 8$, compared with 1.000 for typed tokens and 0.123 for the shared sum. In raw accuracy the shared sum reached 0.527, 0.327, and 0.233 at $K = 2, 4, 8$, close to the bounds above, so it learned nearly all that the histogram allows. Our disjoint construction pays in width, allocating A coordinates per track ($D \geq KA$, tested at $D = 128 = KA$); this is a property of the construction rather than a lower bound for binding-preserving additive codes. What matters is that each track has its own code, and near-orthogonal random codes trade exact separation for width, since the number of value–type pairs a fixed-width superposition can hold without interference grows with width (Clarkson et al., 2023). Per-channel input encoders are one such code, and in Section 5 we show that on real data they change which fusion method wins. Together, these results place the failure in the encoding rather than in the act of summing.

We summarize these findings as a single principle, *bind before you compress*: keep the distinctions a selective target may need until a routing step has used them, and only then compress. This is a statement about ordering rather than a prohibition. Compression can come early whenever the encoding already preserves the needed pairing, or when the pairing is irrelevant to the task.

3 CONTROLLED EVALUATION

To test when binding matters in practice, we designed a set of synthetic tasks that share one frame and differ in a single factor (Table 1, Fig. 1d–g). Each task presents a sequence of L positions, where each position carries a primary token and up to $T - 1$ modifier values. The model reads the whole sequence in both directions and must output a value at a query position. What varies between tasks is *how* the query names the value it wants, whether by position, by content, or by track, and this determines whether summing modifiers at the input destroys the answer. Only the track-addressed task instantiates Theorem 1 exactly. The other two use free embeddings, so the summed input remains identifiable in principle, and any failure there reflects a difficulty of training rather than missing information.

Table 1: The synthetic tasks. Each isolates one way of naming the value a query wants; normalized accuracy is measured against each task's own reference accuracy and never compared across rows.

Task	Input → target	What summing the inputs loses
Position addressing (Fig. 1d)	Position i holds a value plus unrelated extra values (their number is the *crowding depth*); positions are paired by a fixed contact map. At j, output the value written at its contact i.	The wanted value is mixed with the extras at i and must be recovered before it can be carried to j.
Content addressing (Fig. 1e)	Key–value pairs sit at scattered positions with keys drawn afresh each example. At a query carrying one key, output the value stored under that key.	As above, and the match is by content rather than position (Fu et al., 2023; Arora et al., 2024a).
Track addressing (Fig. 1f)	One position holds K values, one per named track, encoded by one shared value table. Given a cue naming a track, output its value.	Theorem 1 exactly: the sum does not record which track held which value, so accuracy is capped at the histogram ceiling.
Aggregation (Fig. 1g)	The same K-value bundle. Output the maximum, the count of a value, or the number of distinct values.	Nothing that matters: the targets depend only on the multiset of values. Used as controls.
Sparse relevance (§4.2)	Position addressing at useful depth one plus *live* crowders (informative table) or *dead* crowders (never-informative table). Output the one useful value.	Live crowders look like the useful value to an encoder blind to their track, so they interfere although only one item matters.
Mixed (§4.1)	A bundle whose tracks feed an aggregation target and a retrieval target; output both.	Only the retrieval half, which the hybrid exploits.

In every comparison, the two fusion methods see the same geometry or layout and the same seeds. We report normalized accuracy $q = (a - a_0)/(1 - a_0)$, where a_0 is a reference accuracy declared by each task (Appendix B), so that 0 marks the reference and 1 a perfect score. A run counts as solved when it crosses a declared threshold, 0.5 normalized accuracy unless stated (55% raw accuracy on the alphabet-10 contact task). For the contact-map experiments, the map is the unit of replication, with seeds nested within each map. Failed runs are retained in all summaries, and Appendix B gives the metrics and hyperparameters of every experiment.

3.1 Load and training budget

We first asked how summed fusion copes as more content is packed into each position. On five fresh contact maps at 4,000 steps, factored fusion solved position addressing at every crowding depth we tested, while summed fusion degraded from certain success at depth 1 to 0.188 at depth 4 and failed entirely at depth 8 (Fig. 2a). Part of this gap is a training effect. Tripling the budget shrank the depth-4 gap in solve probability from 0.812 to 0.425 [0.237, 0.613] (Fig. 2b,c), and the summed loss kept falling long after factored fusion had converged (Fig. 2i). The transition is therefore one of trainability, and its location moves with the training budget. A preregistered prediction that it would also scale with embedding width and rank did not hold (Appendix Fig. A1).

To check that the gap is not an artifact of one shared training recipe, we gave each fusion method its own search over twelve recipes, froze the winners, and retrained on five fresh geometries with three seeds each. Factored fusion reached 1.000 and summed fusion -0.003, a paired gap of 1.003 [0.999, 1.006] (Fig. 2d), with the best summed recipe staying near chance despite having more trainable parameters. Transfer was modest for both methods: a model trained on one geometry kept only part of its skill on another (Fig. 2k).

3.2 Pause controls and stage placement

Separate tokens also lengthen the sequence that the recurrence runs over, which could help on its own by giving the model more computation. To separate these two effects, we replaced each modifier token in the summed model with a learned, content-free "pause" token, one per slot (Goyal et al., 2024; Darcet et al., 2024). The pause model therefore sees exactly as many tokens as the factored one, but its values are still added into the primary. At depths 4 and 8, the pause model solved 0.175 and 0 of the time, tracking summed fusion rather than factored fusion (Fig. 2a), and a similar pause ladder in the content-addressed task also failed through 24,000 steps. This suggests that the benefit of factoring comes from the content of the separate tokens rather than from the added length.

We next asked where in the network the typed content is actually used. Under a parameter-matched comparison, we moved the recurrent mixer between stages. With recurrence only in stage 2, after DROP, distance-4 accuracy stayed near zero for both mixers. Moving it to stage 1 raised accuracy to 0.958 for both GDN2 and KDA, and adding a recurrent stage 2 on top gave no further gain (Fig. 2f,g). In this configuration, then, routing helped only while the modifier content was still present, which is where the architecture places it.

Finally, we asked what DROP costs. We compared it with **KEEP**, a variant that retains all LT tokens through stage 2 instead of dropping the modifiers. Linear probes (Fig. 2e) showed the same information path for both variants, although probes report what is linearly recoverable rather than what the model uses (Alain & Bengio, 2017; Belinkov, 2022), so we treat them as descriptive. Held-out accuracy was 0.482 with DROP and 0.480 with KEEP (Fig. 2h). The two therefore had similar observed held-out accuracy, and DROP's contribution is computational: it shrinks the downstream sequence from LT to L tokens.

4 Addressing tasks, aggregation tasks, and hybrid fusion

Having established the effect on position addressing, we next tested whether it holds when the query names its value in other ways (Fig. 3a). Content-addressed retrieval was solved by both fusions at low load, but at load 8 summed fusion remained below the task's selection reference through 24,000 steps. Track-addressed retrieval under a shared value map behaved as the histogram ceiling predicts, with factored input solving the task through eight tracks. Although the three tasks fail for different

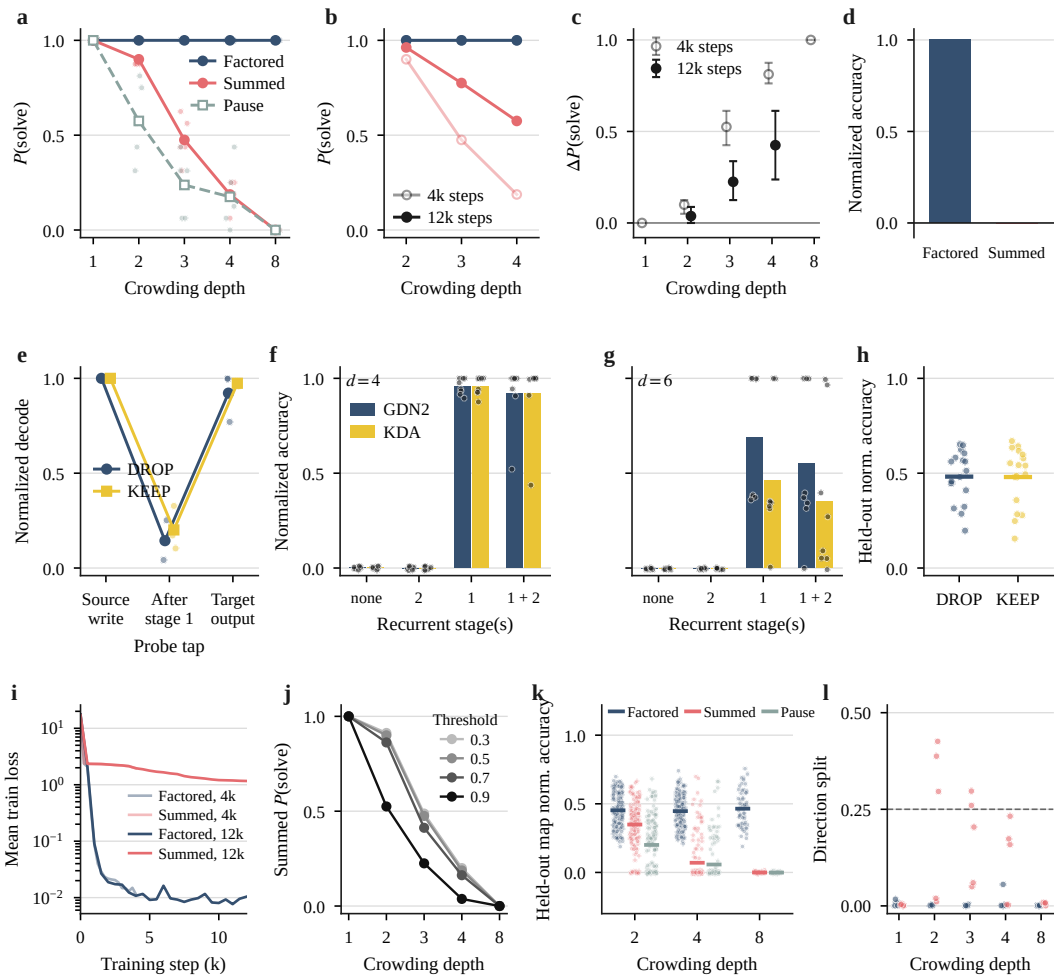


Figure 2: Controlled evidence. Top, five fresh contact maps: (a) solve probability versus crowding depth at 4k steps, with length-matched pause controls. Dots are maps. (b) Summed transition at 4k versus 12k steps. (c) Paired factored−summed gap with five-map bootstrap intervals. (d) Held-out accuracy after independent per-fusion recipe search. Middle, mechanism: (e) probe decode along the path for DROP and KEEP ($n = 3$ maps). (f, g) Stage-placement intervention at distance 4 and 6, eight seeds per bar. (h) DROP versus KEEP held-out accuracy ($n = 18$ each). Bottom, contact-task details: (i) mean training loss per budget and fusion. (j) Summed solve probability at thresholds 0.3–0.9. (k) Accuracy on held-out geometries at depths 2, 4, 8 (seed $\times$ map units, bars are means). (l) Per-map direction split at 4k steps, dashed line at 0.25.

reasons, they share the same symptom: summed fusion struggles whenever the answer depends on which input held which value.

The aggregation controls show the other side of this boundary (Fig. 3b). On maximum, count, and set-size targets, neither fusion held a robust final-accuracy advantage, and summed fusion reached the set-size threshold sooner (median 250 versus 750 steps). Both failed a modular-addition target. This fits the theorem: an aggregate does not depend on which track held which value, so the assignment obstruction does not apply, although a pooled encoding must still keep enough of the multiset to compute it (Zaheer et al., 2017; Murphy et al., 2019), even though the task may still be hard to learn.

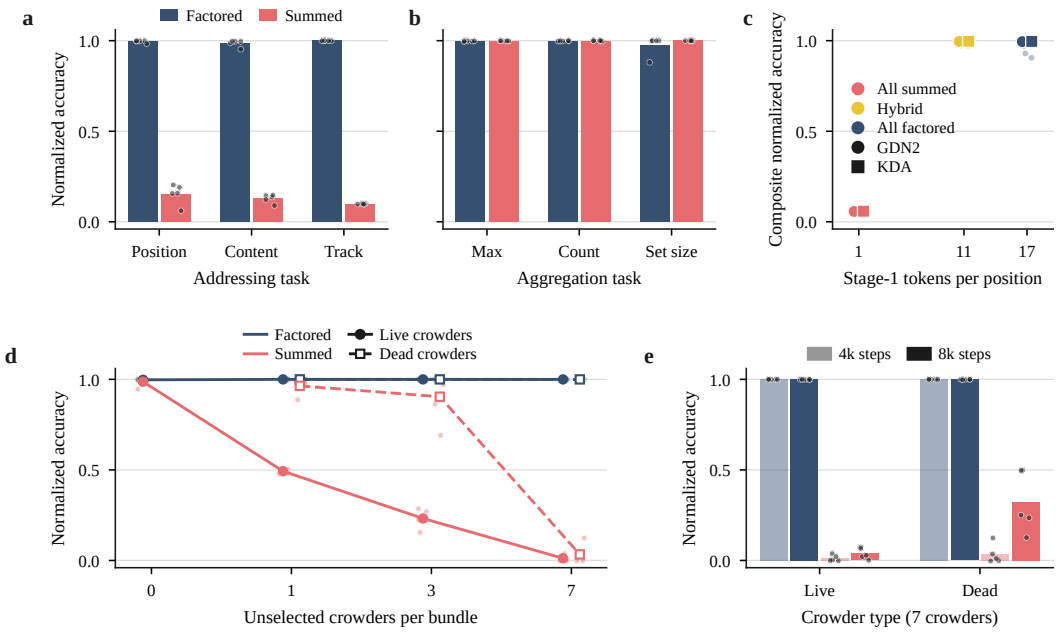


Figure 3: Task boundary and sparse relevance. (a) Addressing tasks at load 4 under GDN2 (the load-8 content result in the text is not plotted) and (b) aggregation tasks. Bars are means over five geometries or layouts, dots are units, and scores are not comparable across families. (c) Mixed aggregation/retrieval construction: composite accuracy versus stage-1 tokens per position for all-summed, hybrid, and all-factored under GDN2 (circles) and KDA (squares). Large markers are means over 40 arms. (d) Accuracy versus number of unselected crowders, live (solid) versus dead (dashed). Dots are layout means. (e) Budget check at seven crowders, 4k versus 8k steps.

4.1 Hybrid fusion

If some inputs are safe to pool and others are not, a natural next step is to pool selectively. Let $\mathcal{A}$ be the inputs an aggregation target consumes and $\mathcal{R}$ the inputs a selective target needs. The **hybrid** builds its primary token from the sum over $\mathcal{A}$ and exposes only $\mathcal{R}$ as typed slots to stage 1, with the partition specified by hand. In the mixed task of Table 1, pooling everything hurt both targets, factoring everything solved both, and the hybrid solved both with fewer tokens (Fig. 3c). Composite scores were 0.995 (all-factored) versus 0.996 (hybrid) under GDN2 and 0.996 versus 0.997 under KDA, while stage-1 tokens per position fell from 17 to 11, a 35% reduction at matched parameters. The hybrid therefore bought a shorter expanded representation at similar observed quality, although the wall-clock saving is smaller than the token saving and pure summation remains the shortest stream.

4.2 Sparse relevance

When only one item matters, it can seem safe to pool the rest away. To test this, we fixed the useful depth at one in the position-addressed task and varied what else was written alongside it. **Live crowders** share the informative value table and could, in another context, be the item that matters, so an encoder that is blind to their track cannot suppress them selectively. **Dead crowders** draw from a table that is never informative and can, in principle, be learned away. Unlike in the missing-modality setting (Neverova et al., 2016; Havaei et al., 2016; Mckinzie et al., 2023), the whole bundle is always present.

At 4,000 steps, summed accuracy under live crowders fell from 0.987 to 0.493, 0.232, and 0.012 with one, three, and seven crowders, while factored fusion stayed near one (Fig. 3d). Doubling the budget barely moved the hardest live cell (0.012 to 0.038), whereas dead crowders were partly learned away (0.033 to 0.321, Fig. 3e). This suggests that suppressing irrelevant content is a matter of budget when the content is recognizably irrelevant, and a matter of binding when it is not.

5 REAL-DATA COMPARISONS

Section 2 locates the obstruction in a shared encoder; a per-track encoder removes it, though nothing then guarantees that summation does well. We asked whether the ranking of the two fusions follows the encoder on real data. For weather we compared them at equal wall-clock time on matched hardware rather than equal steps, because summed models take far more optimizer steps per second. The models are small research configurations, far from operational systems (Lam et al., 2023; Bi et al., 2023).

Weather forecasting. On WeatherBench 2 (Rasp et al., 2024) at 2.8° resolution we predict Z500 six hours ahead from all 69 ERA5 fields at each grid cell, with Z500 as the primary and the other 68 fields as modifier tracks (Fig. 4a); per-variable tokenization of a grid cell has a precedent in ClimaX (Nguyen et al., 2023). Each arm trains for 2640 s of wall clock on one GPU, with recipes chosen on a development split and frozen, five seeds, and best-validation checkpoints scored on a held-out test year. With **per-channel encoders**, the common practice in which every field has its own input projection, summed fusion wins clearly: 89.4 ± 1.0 m²/s² test RMSE against 137.4 ± 2.2 for the hybrid and 150.3 ± 3.6 for factored (Fig. 4c, left). It does so while completing about 16× more optimizer updates in the same time. With a **shared encoder**, one small nonlinear value encoder for all 69 fields plus an additive type embedding, which is the setting of Theorem 1, the ranking reverses: factored reaches 146.0 ± 4.7 and summed only 249.2 ± 8.8 (Fig. 4c, right), even though summed again takes about 12× more updates. Step count does not explain the gap: summed's first checkpoint, at about 3.2k updates (more than factored completes in its whole budget), is already about 2.8× worse than factored's last, and it never catches up (Fig. 4d). The nonlinear encoder does not restrict pooling to a scalar total, but the pooled vector is unchanged when values are reassigned across fields; both arms share the encoder and parameter count (Appendix E). Both shared-encoder arms were still improving at the cap, so the size of the gap is specific to this budget and search.

Land-cover classification. BigEarthNet-19 (Sumbul et al., 2019; 2021; Clasen et al., 2025) is closer to an aggregation task: nineteen land-cover labels are predicted for a whole Sentinel-1/2 patch from twelve optical bands and two radar polarizations, with the red band as the primary (Fig. 4b). We trained 14M-parameter factored and summed models under both encoder conditions, three seeds each, with early stopping on validation macro-AP rather than a common time budget, and scored macro-AP on a fixed 40,000-patch test subsample. The same interaction appears (Fig. 4e). With per-channel encoders the two fusions scored similarly (0.708 ± 0.002 factored versus 0.710 ± 0.003 summed). With a shared encoder factored leads by 1.9 percentage points (0.714 ± 0.004 versus 0.695 ± 0.001). Because the per-channel encoders are also deeper, the contrast reflects sharing and encoder depth together. Dedicated Sentinel-1/2 fusion models (Hoffmann et al., 2023; Fuller et al., 2023; Houdre et al., 2025; Gupta et al., 2025; Wang et al., 2025) or remote-sensing foundation models (Guo et al., 2024; Zhang et al., 2025) reach higher absolute scores.

Read together, the two datasets are consistent with the mechanism: what matters is less addition versus tokens than whether track identity survives to the point where the task needs it. When the encoder bound it, summing was cheap and effective; when it did not, separate tokens recovered what extra training steps within the budget did not.

A competitive alternative. Primary-query cross-attention is another way to bind before compressing (Lee et al., 2019). On the synthetic tasks it passes the positive controls the dense baseline failed, and at length 256 no robust advantage for factored routing was established. With recipes frozen at 256 it trained less reliably at 1024 and 2048 (6/15 and 1/15 successes against 15/15), a reliability result rather than a retuned comparison (Appendix D).

6 RELATED WORK AND LIMITATIONS

Sum-based set representations suit functions that should not care about order (Zaheer et al., 2017; Wagstaff et al., 2019; Murphy et al., 2019); preserving a role–value association is a different requirement (Smolensky, 1990; Chen et al., 2020). Attention bottlenecks and input-to-latent compression are well established (Nagrani et al., 2021; Jaegle et al., 2021; 2022; Li et al., 2023), as are token exchange between modality streams and compression of aligned chunks (Wang et al., 2022; Piergiovanni et al., 2024). Two already bind before compressing: Perceiver attaches position and modality features to each input before cross-attending into a latent array, and ClimaX (Nguyen et al., 2023)

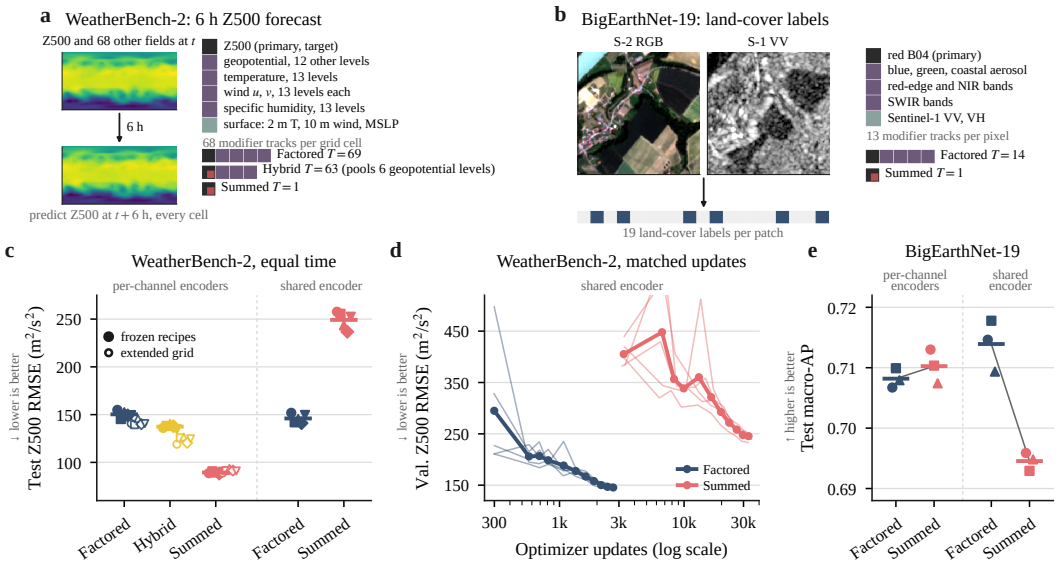


Figure 4: Real-data comparisons. (a) WeatherBench 2: from all 69 ERA5 fields at time t on the 2.8° grid (maps: Z500 on 2016-07-01 00 UTC and six hours later), predict Z500 at $t + 6$ h at every cell. Purple tracks are pressure-level fields and grey tracks surface fields; token rows give stage-1 tokens per position and a red inset marks tracks pooled into the primary. (b) BigEarthNet-19: from a Sentinel-2 (RGB) and Sentinel-1 (VV) patch, predict nineteen labels; highlighted boxes are this patch's six. (c) Test Z500 RMSE after 2640 s of training, five seeds (text gives mean $\pm$ s.d.), with per-channel encoders (left) or one shared encoder (right). Filled markers: frozen recipes (recomputed from arm-level records in a separate session); hollow: a second recipe grid (not recomputed); bars are means. (d) Validation Z500 RMSE against optimizer updates (log scale) for the shared-encoder arms of (c); faint lines are seeds, bold lines seed means. (e) BigEarthNet macro-AP for the four arms, trained to best validation, three seeds each; grey lines join fusion means within an encoder condition.

tokenizes weather variables separately and cross-attends over them at each location before spatial processing. Tokaformer differs in mechanism, a recurrent pass over the typed stream and a fixed gather, and we offer the principle as a way to read all three. Adaptive token reduction (Ryoo et al., 2021; Bolya et al., 2023) chooses which tokens to drop or merge; DROP is fixed rather than adaptive because the typed tokens have already been consumed by routing when they are removed; combining routing with adaptive merging is a natural extension. Linear-recurrent backbones have also been used to build multimodal models (Zhao et al., 2025; Qiao et al., 2024), where the recurrence replaces attention over a concatenated stream; here it plays a narrower role, routing over contiguous typed bundles before compression. Position-local operators such as FiLM, low-rank fusion, and MMTM (Perez et al., 2018; Liu et al., 2018; Joze et al., 2020) mix modalities inside one vector per position and form the baseline panel in Appendix C. Work on the theory and optimization of multimodal learning (Huang et al., 2021; Lu, 2023; Wang et al., 2020; Peng et al., 2022; Kontras et al., 2025) shares our reason for separating what a representation cannot express from what a model struggles to learn.

The evidence has limits. Most of it is synthetic and bidirectional, at budgets of a few thousand steps, with modest transfer to held-out geometries, and the hybrid's partition is chosen, not learned. The real-data comparisons are finite-budget observations at one resolution and model size each; the shared-encoder summed arm was not tuned beyond the edge of its grid; and on BigEarthNet the encoder conditions differ in depth as well as sharing. Comparisons with adaptive token compression and with dense Transformer and state-space backbones at matched quality remain open. The working rule, then, is to bind before you compress, and to pool wherever the encoding already keeps track identity or selection is unnecessary.

REPRODUCIBILITY STATEMENT

The appendix specifies the formal assumptions, experimental units, controls, metrics, and known provenance limitations. The accompanying author-facing artifact package retains hash-pinned inputs, generated statistical-unit tables, and deterministic figure builds; the manuscript's headline numerical fields are resolved from those same facts. Report-level evidence is explicitly distinguished from direct arm-level recomputation.

AI USE STATEMENT

Generative AI tools assisted research planning, hypothesis and experiment design, implementation, formal exposition, result analysis and interpretation, verification workflows, and manuscript and artifact preparation. The verification history includes separate producer and verifier sessions; this separation is a process check and is not human validation. This draft consolidates existing experimental results without running new training experiments. The authors remain responsible for the final claims, citations, artifacts, and disclosure.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (ICLR), Workshop Track*, 2017. URL https://arxiv.org/abs/1610.01644.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2312.04927.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of ICML*, 2024b. URL https://arxiv.org/abs/2402.18668.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2): 423–443, 2019. doi: 10.1109/TPAMI.2018.2798607.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. doi: 10.1162/coli_a_00422.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619:533–538, 2023. doi: 10.1038/s41586-023-06185-3.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Kezhen Chen, Qiuyuan Huang, Hamid Palangi, Paul Smolensky, Ken Forbus, and Jianfeng Gao. Mapping natural-language problems to formal-language solutions using structured neural representations. In *Proceedings of ICML*, 2020. URL https://proceedings.mlr.press/v119/chen20g.html.

Kenneth L. Clarkson, Shashanka Ubaru, and Elizabeth Yang. Capacity analysis of vector symbolic architectures, 2023. URL https://arxiv.org/abs/2301.10352.

Kai Norman Clasen, Leonard Hackel, Tom Burgert, Gencer Sumbul, Begum Demir, and Volker Markl. reBEN: Refined bigearthnet dataset for remote sensing image analysis. In *IEEE International Geoscience and Remote Sensing Symposium*, 2025. URL https://arxiv.org/abs/2407.03653.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.16588.

fla-org contributors. Flash linear attention. Software repository, 2026. URL https://github.com/fla-org/flash-linear-attention. Pin exact release/commit used in experiments.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2212.14052.

Anthony Fuller, Koreen Millard, and James R. Green. Croma: Remote sensing representations with contrastive radar-optical masked autoencoders. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/11822e84689e631615199db3b75cd0e4-Abstract-Conference.html.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.02226.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks, 2020. URL https://arxiv.org/abs/2012.05208.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2312.00752.

Xin Guo, Jiangwei Lao, Bo Dang, Yingying Zhang, Lei Yu, Lixiang Ru, Liheng Zhong, Ziyuan Huang, Kang Wu, Dingxiang Hu, Huimei He, Jian Wang, Jingdong Chen, Ming Yang, Yongjun Zhang, and Yansheng Li. Skysense: A multi-modal remote sensing foundation model towards universal interpretation for earth observation imagery. In *Proceedings of CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Guo_SkySense_A_Multi-Modal_Remote_Sensing_Foundation_Model_Towards_Universal_Interpretation_for_Earth_Observation_Imagery_CVPR_2024_paper.html.

Debashis Gupta, Aditi Golder, Rongkhun Zhu, Kangning Cui, Wei Tang, Fan Yang, Ovidiu Csillik, Sarra Alaqahtani, and V. Paul Pauca. MoSAiC: Multi-modal multi-label supervision-aware contrastive learning for remote sensing, 2025. URL https://arxiv.org/abs/2507.08683.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention, 2026. URL https://arxiv.org/abs/2605.22791.

Mohammad Havaei, Nicolas Guizard, Nicolas Chapados, and Yoshua Bengio. HeMIS: Hetero-modal image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2016. URL https://arxiv.org/abs/1607.05194.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025. doi: 10.1126/science.ads0018.

David Hoffmann, Kai Norman Clasen, and Begum Demir. Transformer-based multi-modal learning for multi-label remote sensing image classification. In *IEEE International Geoscience and Remote Sensing Symposium*, 2023. doi: 10.1109/IGARSS52108.2023.10281927. URL https://arxiv.org/abs/2306.01523.

Nicolas Houdre, Diego Marcos, Dino Ienco, Laurent Wendling, Camille Kurtz, and Sylvain Lobry. Promm-rs: Exploring probabilistic learning for multi-modal remote sensing image representations. In *WACV Workshops*, 2025. doi: 10.1109/WACVW65960.2025.00063. URL https://openaccess.thecvf.com/content/WACV2025W/GeoCV/html/Houdre_ProMM-RS_Exploring_Probabilistic_learning_for_Multi-Modal_Remote_Sensing_Image_Representations_WACVW_2025_paper.html.

Yu Huang, Chenzhuang Du, Zihui Xue, Xuanyao Chen, Hang Zhao, and Longbo Huang. What makes multi-modal learning better than single (provably). In *Advances in Neural Information Processing Systems*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5aa3405a3f865c10f420a4a7b55cbff3-Abstract.html.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of ICML*, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver io: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.01032.

Hamid Reza Vaezi Joze, Amirreza Shaban, Michael L. Iuzzolino, and Kazuhito Koishida. Mmtm: Multimodal transfer module for cnn fusion. In *Proceedings of CVPR*, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Joze_MMTM_Multimodal_Transfer_Module_for_CNN_Fusion_CVPR_2020_paper.html.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of ICML*, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Kimi Team. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Denis Kleyko, Mike Davies, E. Paxon Frady, Pentti Kanerva, Spencer J. Kent, Bruno A. Olshausen, Evgeny Osipov, Jan M. Rabaey, Dmitri A. Rachkovskij, Abbas Rahimi, and Friedrich T. Sommer. Vector symbolic architectures as a computing framework for emerging hardware. *Proceedings of the IEEE*, 2022. URL https://arxiv.org/abs/2106.05268.

Konstantinos Kontras, Thomas Strypsteen, Christos Chatzichristos, Paul Liang, Matthew B. Blaschko, and Maarten De Vos. Balancing multimodal training through game-theoretic regularization. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f5784de266e45bd09333721b0a602b61-Abstract-Conference.html.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of ICML*, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of ICML*, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of ACL*, 2018. doi: 10.18653/v1/P18-1209. URL https://aclanthology.org/P18-1209/.

Zhou Lu. A theory of multimodal learning. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.52202/075280-2501. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b316495425d076b4abffc065a64c2cca-Abstract-Conference.html.

Brandon Mckinzie, Vaishaal Shankar, Joseph Yitan Cheng, Yinfei Yang, Jonathon Shlens, and Alexander T. Toshev. Robustness in multimodal learning under train-test modality mismatch. In *Proceedings of ICML*, 2023. URL https://proceedings.mlr.press/v202/mckinzie23a.html.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1811.01900.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In *Advances in Neural Information Processing Systems*, 2021. URL https://papers.neurips.cc/paper_files/paper/2021/hash/76ba9f564ebbc35b1014ac498fafadd0-Abstract.html.

Natalia Neverova, Christian Wolf, Graham Taylor, and Florian Nebout. Moddrop: Adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. doi: 10.1109/TPAMI.2015.2461544. URL https://arxiv.org/abs/1501.00102.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. In *International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/nguyen23a.html.

Xiaokang Peng, Yake Wei, Andong Deng, Dong Wang, and Di Hu. Balanced multimodal learning via on-the-fly gradient modulation. In *Proceedings of CVPR*, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Peng_Balanced_Multimodal_Learning_via_On-the-Fly_Gradient_Modulation_CVPR_2022_paper.html.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of AAAI*, 2018. doi: 10.1609/aaai.v32i1.11671. URL https://ojs.aaai.org/index.php/AAAI/article/view/11671.

AJ Piergiovanni, Isaac Noble, Dahun Kim, Michael S. Ryoo, Victor Gomes, and Anelia Angelova. Mirasol3b: A multimodal autoregressive model for time-aligned and contextual modalities. In *Proceedings of CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Piergiovanni_Mirasol3B_A_Multimodal_Autoregressive_Model_for_Time-Aligned_and_Contextual_Modalities_CVPR_2024_paper.pdf.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3): 623–641, 1995. doi: 10.1109/72.377968.

Yanyuan Qiao, Zheng Yu, Longteng Guo, Sihan Chen, Zijia Zhao, Mingzhen Sun, Qi Wu, and Jing Liu. VL-Mamba: Exploring state space models for multimodal learning, 2024. URL https://arxiv.org/abs/2403.13600.

Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*, 2024. URL https://arxiv.org/abs/2308.15560.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Imanol Schlag, Kazuki Irie, and Juergen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of ICML*, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1–2):159–216, 1990. doi: 10.1016/0004-3702(90)90007-M.

Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. BigEarthNet: A large-scale benchmark archive for remote sensing image understanding. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 5901–5904, 2019. doi: 10.1109/IGARSS.2019.8900532.

Gencer Sumbul, Arne de Wall, Tristan Kreuziger, Filipe Marcelino, Hugo Costa, Pedro Benevides, Mário Caetano, Begüm Demir, and Volker Markl. BigEarthNet-MM: A large-scale, multimodal, multilabel benchmark archive for remote sensing image classification and retrieval. *IEEE Geoscience and Remote Sensing Magazine*, 9(3):174–180, 2021. doi: 10.1109/MGRS.2021.3089174.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of ACL*, 2019. doi: 10.18653/v1/P19-1656. URL https://aclanthology.org/P19-1656/.

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Michael A. Osborne, and Ingmar Posner. On the limitations of representing functions on sets. In *Proceedings of ICML*, 2019. URL https://proceedings.mlr.press/v97/wagstaff19a.html.

Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *Proceedings of CVPR*, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Wang_What_Makes_Training_Multi-Modal_Classification_Networks_Hard_CVPR_2020_paper.html.

Yi Wang, Zhitong Xiong, Chenying Liu, Adam J. Stewart, Thomas Dujardin, Nikolaos Ioannis Bountos, Angelos Zavras, Franziska Gerken, Ioannis Papoutsis, Laura Leal-Taixe, and Xiao Xiang Zhu. Towards a unified copernicus foundation model for earth vision. In *Proceedings of ICCV*, 2025. URL https://arxiv.org/abs/2503.11849.

Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. Multimodal token fusion for vision transformers. In *Proceedings of CVPR*, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Wang_Multimodal_Token_Fusion_for_Vision_Transformers_CVPR_2022_paper.html.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of EMNLP*, 2017. doi: 10.18653/v1/D17-1115. URL https://aclanthology.org/D17-1115/.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.

Yingying Zhang, Lixiang Ru, Kang Wu, Lei Yu, Lei Liang, Yansheng Li, and Jingdong Chen. SkySense V2: A unified foundation model for multi-modal remote sensing. In *Proceedings of ICCV*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/papers/Zhang_SkySense_V2_A_Unified_Foundation_Model_for_Multi-modal_Remote_Sensing_ICCV_2025_paper.pdf.

Han Zhao, Min Zhang, Wei Zhao, Pengxiang Ding, Siteng Huang, and Donglin Wang. Cobra: Extending Mamba to multi-modal large language model for efficient inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

A FORMAL DETAILS

Theorem 1 is narrower than it may first appear, and the narrowness is the point. It holds the present-track set, the primary and position contributions, the cue, and everything else the decoder can see fixed, and it asks only what happens when values are exchanged among tracks. The shared value encoder must not depend on which track it is encoding, and the reduction must be commutative in exact arithmetic. Change any of these and the collision can disappear. A track-specific encoder, a joint role–value map, a query-aware map applied before the sum, or a side channel that carries the assignment all defeat it. So the theorem is a statement about one specified representation, not about every architecture that contains an addition.

The accuracy ceiling follows from a symmetry argument, using the cue J of the main text. Conditional on the histogram $H = h$ of values, exchangeability makes every assignment consistent with h equally likely, so $P(v_J = a \mid H = h, J) = h_a/K$. The best a decoder can do is name a modal value, with conditional accuracy $\max_a h_a/K$, and taking the expectation over H gives the stated ceiling. When the sum does not even identify H, the expression is an upper bound rather than an attainable value. Under uniform sampling with replacement and $A = 16$, the exact ceilings are $17/32$, $1381/4096$, and $69219991/268435456$ at $K = 2, 4, 8$. Sampling without replacement gives exactly $1/K$. A randomized decoder mixes conditional label probabilities and cannot exceed their maximum, so randomness does not help. For dependent or nonuniform values the ceiling becomes $\mathbb{E}[\max_y P(Y = y \mid F(V), Q)]$ and the numbers above no longer carry over. Missing or corrupted tracks likewise change the conditioning, which is why the robustness comparison below is an empirical one.

Typed tokens escape the theorem for a simple reason. If track labels are recoverable and each track's encoding is injective, two inputs with equal representations must agree on every track's value. This proves that the input-level quotient is avoided. It proves nothing about whether learned routing and DROP preserve what the task needs, or whether an efficient optimization path exists. We keep repeating this because it is easy to slide from "the information is there" to "the model will use it." The permutation-invariant targets make the converse point. Maximum, count, set size, and modular sum are functions of the value multiset, so any encoding that identifies the multiset removes the obstruction entirely. The fact that modular addition still fails to train end-to-end shows that identifiability is not an optimization guarantee.

The disjoint code places each track's value table in its own orthogonal channel subspace and needs $D \geq KA$ dimensions. At $K = 8, A = 16$ we used $D = 128$. The frozen per-track tables add stored parameters but not trainable ones. The experiment used GDN2 with five layouts and two seeds per cell, so it is a demonstration that the construction works rather than a replicated comparison across mixers, and the width condition is a property of this construction rather than a lower bound for every binding-preserving encoder. Finally, the ideal invariance is not bitwise invariance in float32. We took the trained summed-fusion model, swapped values between tracks, and compared the fused vectors: every tested permutation agreed within tolerance, none bit-for-bit, with a maximum residual of $5.7220458984375 \times 10^{-6}$. Roundoff is a small numerical side channel, and the impossibility statement applies to the ideal commutative observation.

B TASK FAMILIES, METRICS, AND PROTOCOL

The table below gives the construction, budget, and replication of every experiment in the paper, with the figure panel each one feeds. Statistical units are set per family. Multiplying maps by seeds does not turn seeds into independent geometries.

Notation used in the tables: A is the alphabet size (number of distinct values), K the number of tracks, L the sequence length, T the stage-1 tokens per position, C the number of co-located values per write in the content-addressed task, R the routing distance between a write and its query, and D the number of candidate writes a query must choose among (so $1/D$ is that task's chance reference). Appendix A uses D once more, with the explicit qualifier "dimensions", for the width of the disjoint code. Stage-1 mixers: GDN2 and KDA are the two delta-rule linear-attention mixers of Section 2; "banded" is local windowed attention of the same width, used only as a placement control; "KDA slice" means the subset of cells that was replicated with KDA.

Experiment	Construction	Replication	Figure
Contact load and budget	Position addressing on contact geometries, free embeddings, GDN2, depths 1/2/3/4/8 at 4k steps and 2/3/4 at 12k	Five fresh maps, 16 seeds per map at transition depths and six at endpoints	2a–c, i–l
Independent recipe search	Contact depth 8, distance 2, alphabet 10, with twelve recipes per fusion selected on a development geometry, then frozen	Five fresh geometries $\times$ three seeds	2d
Disjoint-code control	Track retrieval with shared, disjoint, and factored encodings, $K = 1, 2, 4, 8$, $A = 16$, $D = 128$	Five layouts $\times$ two seeds per cell	1c
Stage placement	Recurrent mixer in neither stage, stage 2, stage 1, or both, at distances 4 and 6, GDN2 and KDA	Eight seeds per placement and mixer	2f,g
KEEP versus DROP with probes	Contact task with and without modifier removal, linear probes at three taps	Held-out seed $\times$ map units nested in three training maps	2e,h
Content addressing	Bidirectional multi-query recall, $R = 4$, $D = 2$, shared rank-8 value space	Five layouts $\times$ eight seeds, with budget and pause ladders to 24k	3a
Track addressing	Shared-map track retrieval, $K = 1, 2, 4, 8$, $A = 16$	Five layouts $\times$ eight seeds $\times$ two mixers (640 arms)	3a
Aggregation controls	Selective retrieval, learnable max, count, set size, modular addition, mixed targets	Five layouts	3b
Sparse relevance	Useful depth 1 with 0/1/3/7 live or dead crowders, 4k steps and an 8k rerun of the hardest cell	Five layouts $\times$ eight seeds per cell (560 core and 160 budget runs)	3d,e
Hybrid partition	Mixed aggregation/retrieval with a specified partition, 4k steps	Five layouts $\times$ eight seeds $\times$ three fusions $\times$ two mixers	3c
Cross-attention comparator	Positive-control-qualified primary-query attention with frozen recipes, 4k steps	Five held-out layouts $\times$ three seeds	text
Length reliability	Lengths 1024 and 2048 under recipes selected at 256, then a bounded search at 2048	Five layouts $\times$ three seeds per arm and length, then 48 search cells	text
Prospective predictions	Frozen width and rank predictions for the summed crossing, and reach calibration	Held-out crossings, 30 arms per reach cell	A1
Baselines and throughput	Eight position-local fusion modules on the transport task, and warm timing at $L = 1024$	30 arms per fusion on one H100	A2
WeatherBench 2, equal time	Full state $K = 68$ at 2.8°, 2,640 s per run, per-channel encoders (factored, hybrid, summed) and one shared encoder (factored, summed); frozen development recipes	Five seeds per arm and condition; a second recipe grid for the per-channel arms (not independently recomputed)	4a,c,d
BigEarthNet-19, four arms	14M-parameter factored and summed models under per-channel and shared encoders, fixed 8k validation and 40k test subsamples	Three seeds per arm	4b,e
WeatherBench 2, matched steps (earlier)	Per-channel encoders, 3,000 matched steps	Three seeds per fusion	A3a,b

Experiment	Construction	Replication	Figure
BigEarthNet-19, 85M screen (earlier)	Full-data two-arm screen with per-channel encoders, stopped at its time limit	Three seeds per fusion	A3c

Hyperparameters. The recurrent mixers run on Flash Linear Attention 0.5.2 (fla-core 0.5.2), as pinned in the project lockfile. The synthetic experiments share one configuration unless the table says otherwise: width $d_{\text{model}} = 128$, four heads of dimension 32, two stage-1 blocks, feed-forward multiplier 4, learning rate 10^{-3}, batch size 32, gradient-norm clipping at 1.0, and 4,000 optimizer steps. Fusion adds no parameters in any arm, so arms within an experiment are capacity matched by construction, and the stage-1 mixer swap is parameter matched by adjusting the feed-forward width. The factored arm has $T = \text{depth} + 2$ tokens per position on the contact bench and $K + 1$ on the track-addressed bench. Data streams are generated on the fly and depend only on the layout and draw seeds. A dash marks a setting inherited from the bench default that the experiment record does not restate.

Experiment	L	T (factored)	Stage-1 mixer	Stage 2	Steps	Departures from the shared recipe
Contact load and budget	64	3–10	GDN2 (KDA slice)	dense	4k, 12k	alphabet 10, distance 2, embedding rank 4 or free; pause arm at $T = \text{depth} + 2$
Independent recipe search	64	10	GDN2	dense	12k	per-fusion choice of learning rate (3×10^{-4}, 10^{-3}, or 3×10^{-3}), constant or cosine schedule with 10% warmup, and feed-forward multiplier 4 or 6
Disjoint-code control	256	2–9	GDN2	dense	4k	$A = 16$, $D = 128 = KA$ at $K = 8$, frozen per-track value tables for the disjoint arm
Stage placement	64	3	GDN2, KDA, or banded	dense or recurrent	12k	distances 4 and 6; recurrent mixer moved between stages at matched parameters; eight seeds
KEEP versus DROP with probes	64	4, 10	GDN2	dense	4k	KEEP retains all LT tokens through stage 2; linear probes at three reported taps
Content addressing	256	$C + 1$	GDN2, KDA	dense	4k–24k	$R = 4$, $D = 2$, $A = 16$, key vocabulary 64, eight queries, embedding rank 8
Track addressing	256	2–9	GDN2, KDA	dense	4k	$A = 16$, one shared near-orthogonal value table
Aggregation controls	—	2–9	GDN2, KDA	dense	4k–12k	$A = 16$, learnable max, count, set-size, and mixed heads
Sparse relevance	64	2–9	GDN2	dense	4k, 8k	alphabet 10, distance 2, live or dead crowder tables, 0/1/3/7 crowders

Experiment	L	T (factored)	Stage-1 mixer	Stage 2	Steps	Departures from the shared recipe
Hybrid partition	—	17 (11)	GDN2, KDA	dense	4k	fixed partition of tracks into pooled and separate sets; the hybrid has 11 tokens per position
Cross-attention comparator	256	6	GDN2 (3 blocks) or primary-query attention	dense	4k	$C = 4$ and 8, $D = 2$; 12 optimizer recipes and 4 shapes searched per arm on development layouts, then frozen; factored 3×10^{-4} cosine, comparator 10^{-3} constant
Length reliability	1024, 2048	6	GDN2 (3 blocks) or primary-query attention	dense	12k or a common wall cap	recipes frozen at $L = 256$; success threshold 0.9; B200 GPUs, three runs packed per GPU
Prospective predictions	64	3–10	GDN2	dense	4k, 12k, 16k	width 192 with six heads, embedding rank 6, or alphabet 16 for the falsification cells; $T = 3$ versus 10 at distances 5 and 7 for reach
Baselines and throughput	64; 1024	3–10; 1, 8, 16	GDN2, dense attention	dense	4k	eight position-local fusion modules with alphabet 10; warm timing at batch 8 and 32 on one H100 (a Mamba-2 mixer was timed but its quality runs did not complete and are not reported)
WeatherBench 2, equal time	8192	69 (63)	GDN2	recurrent	2,640 s	AdamW, weight decay 0.05, best-validation weights; frozen per-channel recipes on H200 at learning rate 2×10^{-3} (cosine to the cap; hybrid constant after warmup), the second per-channel grid at each arm's development winner (4×10^{-3} or 8×10^{-3}), shared-encoder arms on B200 at 8×10^{-3}, the top of the summed arm's development grid; the hybrid folds six geopotential levels into the primary

Experiment	L	T (factored)	Stage-1 mixer	Stage 2	Steps	Departures from the shared recipe
WeatherBench 2, matched steps	8192	69 (63)	GDN2	recurrent	3k	AdamW, weight decay 0.05, 300 warmup steps, cosine, batch 2; per-channel encoders
BigEarthNet-19, four arms	patches	14	GDN2	—	to best validation	$d_{\text{model}} = 384$ (14M parameters), AdamW, learning rate 10^{-3}, 8k validation and 40k test subsamples; per-channel or shared value encoder with additive type embedding
BigEarthNet-19, 85M screen	patches	14	GDN2	—	15 h wall time	$d_{\text{model}} = 768$, 12 heads, 5 + 5 + 2 blocks; AdamW, weight decay 0.05, 500 warmup steps, cosine, batch 16; per-channel encoders

Metrics. Each generator declares its own reference accuracy a_0, and normalized accuracy is measured against it: $1/A$ for contact tasks, $1/D$ for content retrieval (with $D = 2$, so negative scores are possible and mean worse than the reference), and $1/K$ for track retrieval when $K > 1$. For track retrieval $1/K$ is a selection reference, not the accuracy of random output: guessing a value uniformly gives $1/A$, outputting a random track's value gives $1/K + (K - 1)/(KA)$ under sampling with replacement, and the histogram bound of Appendix A is an upper bound through a shared sum, attained when the fused vector identifies the histogram. At $K = 1$ raw accuracy is reported and the normalized score is undefined. Because the references differ, scores from different families sit on different scales and are never pooled, and where families share an axis (Fig. 3a) the caption states each task's condition and scores are not to be compared across tasks. The hybrid's composite accuracy is the fraction of scored reads at which both the aggregation and the routing outputs are exactly correct, with no weighting; it is normalized against the product of the two component references, the aggregation marginal ceiling (the held-out accuracy of always predicting the most frequent aggregation outcome, estimated from 4,000 generated sequences) and $1/A$ for routing, and the two subscores are reported separately in the records. Weather RMSE is reported in m²/s² of geopotential, latitude-weighted, for a single 6-hour step at 2.8° on a two-year training period. The earlier matched-step runs recorded RMSE on a normalized scale; the recovered normalization is a per-field z-score with training-set statistics, and the Z500 standard deviation of 3368.28 m²/s² converts those values to the physical units used in Fig. A3. These numbers are not comparable to operational leaderboard entries at 0.25° and multi-day leads. BigEarthNet uses macro average precision. The synthetic success threshold is 0.5 normalized accuracy (55% raw accuracy on the alphabet-10 contact task), with sensitivity at 0.3, 0.7, and 0.9 shown in Fig. 2j, and the reliability study uses a stricter 0.9. Convergence flags are kept separate from success, since a model can plateau at chance or keep lowering loss after it has solved the task.

Pairing and geometry. Data streams depend only on the layout and draw seeds, never on the fusion arm, and recorded fingerprints confirm the pairing. Seeds within a layout measure initialization variation. The contact matrix treats the map as the unit and uses a deterministic map bootstrap with 5,000 replicates. With five maps, a consistent sign has an exact two-sided floor of 0.0625, and a saturated interval describes the sample rather than certifying universal success. Where direction within a map is informative we show it separately (Fig. 2l).

Recipes. The contact matrix uses one controlled recipe throughout. The independent search instead lets each fusion choose its own learning rate, schedule, and feed-forward multiplier from twelve candidates on a development geometry, then freezes the choice. Factored selected 10^{-3}, cosine,

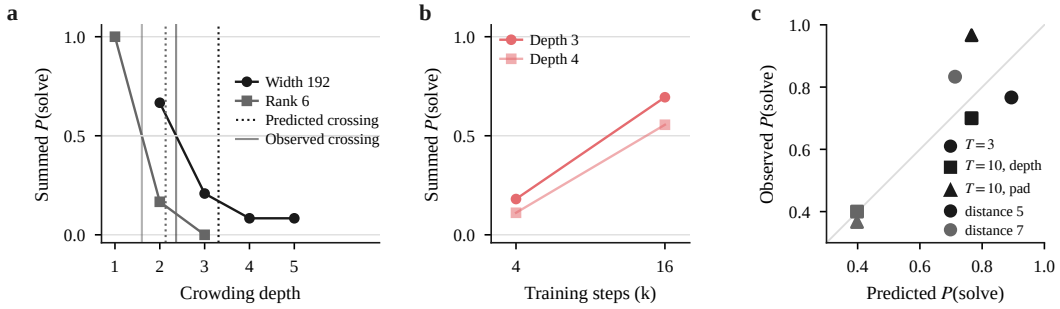


Figure A1: Prospective predictions. (a) Summed solve probability versus depth for the width-192 and rank-6 cells, with the frozen predicted crossing (dotted) and observed crossing (solid). (b) Summed solve probability at 4k versus 16k steps for depths 3 and 4. (c) Reach calibration, observed versus predicted solve probability ($n = 30$ arms per cell).

multiplier 4 (about 1.11M trainable parameters), and summed selected 3×10^{-3}, cosine, multiplier 6 (about 1.44M). This addresses a finite tuning concern, not every optimizer or curriculum.

C Additional controls and boundary checks

A prediction that failed. Early in the program we proposed that the summed transition depth should scale with embedding width and rank, and we froze the prediction before testing it. It was wrong in direction. The width-192 crossing was predicted at depth 3.31 and observed at 2.36, the rank-6 crossing was predicted at 2.13 and observed at 1.60, and added training budget moved the boundary in the expected direction (Fig. A1a,b). We report this because it is the reason the main text describes a budget-sensitive trainability transition rather than a capacity law. The reach calibration in Fig. A1c tells a related story. Expanding stage-1 tokens from three to ten per position at fixed content lowered distance-8 performance from 0.6508 to 0.1578 at 12k steps, and real content load has an effect beyond token count (0.70 versus 0.97 for padding at distance 5). Neither a fixed positional reach nor a token-only reach law holds. Reach depends on tokens, content, geometry, and budget together, which is the practical argument for expanding selectively rather than everywhere.

Transfer and direction. A model trained on one contact geometry and evaluated on another keeps only part of its skill. Factored held-out accuracy at depth 8 is about 0.464 against 1.0 on the training map (Fig. 2k). Factored transfers better than summed, but we would not describe either as generalizing to arbitrary contact graphs. Some summed and pause cells are direction-sensitive in the confirmatory grid while factored is not (Fig. 2l), and the endpoint contrast does not depend on treating seeds as maps.

Robustness. In track retrieval with one independently corrupted non-answer track, factored stays near 1 while summed falls to about 0.036 normalized accuracy. The raw-accuracy gap widens as well, so normalization is not the source of the effect. Factored is stable across the presence shifts we tested. The summed presence slope is confounded, because removing tracks also removes histogram collisions, so we do not read it as a missing-modality result. Calibration on no-answer examples is comparable for both. The licensed reading is corruption isolation in this construction, not a general sensor-robustness guarantee.

Aggregation and the hybrid. "No robust difference" on the aggregation targets is the program's inference rule, not numerical equality. In the displayed set-size slice factored is about 0.976 and summed about 1.000, and the 250-versus-750-step convergence comparison uses coarse observation times. Modular addition is decodable from the summed statistic by a suitable readout yet fails end-to-end, which is the cleanest example we have of identifiability without learnability. The mixed-target conditions contain many nonconverged runs and remain exploratory. The hybrid's token counts come from the actual layouts, including task-query slots. Its comparison is against both a fully factored and a fully summed arm: it beats the first on stream length at similar quality and the second on quality. It does not learn its partition, and its load diagnostics are secondary.

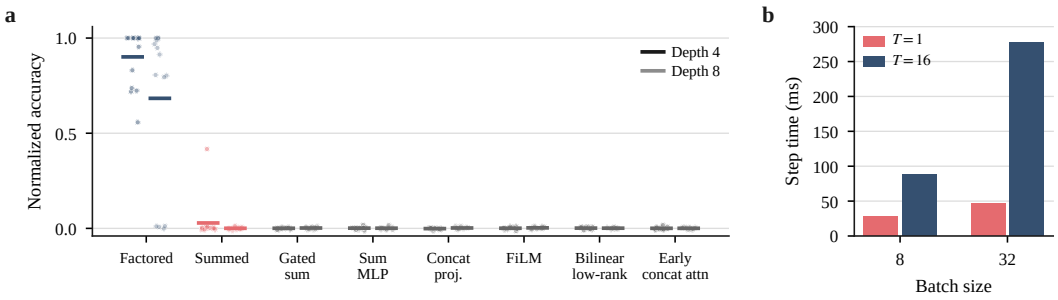


Figure A2: Baselines and systems. (a) Baseline panel at distance 4, depth 4 (dark) and 8 (light), 15 seed/map arms per fusion and depth. Bars are means. The dense early-concatenation attention arm failed its positive control and is not a valid Transformer upper bound. (b) Step time at $L = 1024$ on one H100 for $T = 1$ versus $T = 16$ at batch 8 and 32.

Baselines and throughput. Eight position-local fusion modules, including gated and MLP sums, concatenation with projection, FiLM (Perez et al., 2018), and bilinear low-rank fusion (Liu et al., 2018), were run on the transport task at depths 4 and 8 (Fig. A2a). Only factored routing solves it. The dense early-concatenation attention arm also fails its easy positive control, so it cannot serve as a conventional-Transformer upper bound, and the valid comparator is the primary-query cross-attention discussed below. Expanding to $T = 16$ tokens per position costs 3.11× the step time of $T = 1$ at batch 8 and 5.90× at batch 32 on one H100 at $L = 1024$ (Fig. A2b). This is throughput scaling at one coordinate, not an efficiency frontier.

D THE CROSS-ATTENTION COMPARATOR AND TRAINING RELIABILITY

Attention over a set is the standard alternative to sum pooling (Lee et al., 2019), and cross-attention keeps a memory advantage in line with the recall–memory trade-off of fixed-state recurrent mixers (Jelassi et al., 2024; Arora et al., 2024b). The primary-query attention comparator keeps modifier identity, lets primaries attend as queries over all modifier keys and values, and processes the resulting primary stream with dense attention. Its development controls reach 0.9839 at load 1 and 0.9163 at load 4. It has about 1.25M parameters at load 4 against 1.37M for the factored reference, so the comparison is architecture-near rather than resource-identical. The load-4 factored-minus-attention estimate is positive, but the preregistered leave-one-layout-out rule reduces the conclusion to "no robust superiority established". At load 8 the estimate is +0.0536 [−0.0349, 0.1858]. Neither is an equivalence test, and we report the estimates beside the decision so that both the direction and the uncertainty survive.

The reliability study then asks what happens when recipes selected at length 256 are frozen and both models are trained at 1024 and 2048. Non-successful runs are retained in the paired summaries. At 1024 the corrected median paired quality-at-budget difference is +0.8979 [0.0190, 0.9062] and AUC is +0.1860 [0.1543, 0.2761]. At 2048 they are +0.9028 [0.8916, 0.9067] and +0.3513 [0.3132, 0.4136], with five layout clusters as the inference unit. Factored succeeds in 15/15 runs at each length. Attention succeeds in 6/15 and 1/15. Complete-case time-to-threshold would select only the successful comparator runs and is not a fair efficiency estimate. The runs were packed three per GPU and all reached the shared time cap. Separate uncontended probes put microbatch latency at 72.06 versus 69.54 ms at 1024 and 94.19 versus 110.59 ms at 2048 (factored versus attention), but these do not convert the packed histories into a quality–compute frontier. In inference, attention's peak-memory advantage grows with length, from a factored/attention ratio of 1.07–1.28 at 256 to 3.46–7.72 at 4096, with no out-of-memory boundary found through 4096 and batch 32.

The comparator was then given a bounded length-specific search at 2048: twelve configurations over 1,500, 3,000, and 6,000-step budgets. One of 48 final-rung cells solved, and the best four-cell mean was 0.2865 against the 0.90 bar. A pre-outcome amendment advanced every configuration when a rung was nondiscriminating, the final untouched layouts were never evaluated, and part of the factored tuning was cancelled once its recipe froze. A later, larger development search gave each arm twelve recipes per length at update budgets up to 6,000, without a shared time cap; at

that budget the factored arm crossed the 0.9 bar in every one of its 48 cells at each length, and the attention comparator in 5 of 48 at 1024 and 3 of 48 at 2048. These are development-grid counts, not independent draws of an architectural failure probability, and no frozen-winner final comparison was run, so we treat both searches as engineering outcomes about bounded tuning rather than a completed retuned comparison.

E REAL-DATA DETAILS

Equal-time protocol. Every arm in Fig. 4c,d trains for a fixed 2,640 s of wall clock on one GPU with a schedule that ends at the cap, is checkpointed against the development split during training, and is scored once on the test period with the best-validation weights. Arms differ only in fusion and, across the two conditions, in the input encoder. With **per-channel encoders** each of the 69 fields has its own linear projection into the model width before fusion, so the projection carries the field's identity; summing such projections is a full-rank affine map of the 69 values, and an audit of the trained input map confirmed that it is injective on the input. With a **shared encoder** all 69 fields, including Z500, pass through one value encoder f (a 1×1 convolution to the model width, a GELU, and a second 1×1 convolution) and receive a learned additive type embedding t_c, which is the shared-sum construction of Theorem 1. Summation therefore gives $\sum_c [f(x_c) + t_c] = \sum_c f(x_c) +$ const. Because f is nonlinear and its width (128) exceeds the number of fields (69), the latent width does not impose the dimensional obstruction of Wagstaff et al. (2019); we do not claim that the trained pooled representation is injective on multisets. The sum is, however, unchanged when values are exchanged among fields, so channel assignment is lost exactly as Theorem 1 describes. A linear shared encoder would be a different and weaker test: $\sum_c (wx_c + b)$ collapses the fields to their scalar total and discards the multiset as well as the assignment. The nonlinearity rules out that confound. Both shared-encoder arms use the same f and type embeddings and have identical parameter counts (1.22M), so the comparison varies only whether the encoded fields are summed. The per-channel arms ran on H200 GPUs and the shared-encoder arms on B200 GPUs; each comparison is within a GPU class, so the two conditions should be compared by their within-condition rankings and not by absolute error. The summed arm completes about sixteen (per-channel) and twelve (shared) times as many updates as the factored arm in the same time; Fig. 4d plots validation error against updates for the shared arms. The summed arm's development learning rate in the shared condition was the largest value in its grid, and both shared arms were still improving at the cap.

Weather data. ERA5 from WeatherBench 2 at 2.8° (64 latitudes with poles, 128 longitudes), 69 fields: Z, T, U, V, and Q at 13 pressure levels each, plus 2 m temperature, 10 m winds, and mean sea-level pressure. Training uses 2013–2014 (2,920 six-hourly states), validation 2015 (1,460), and test 2016 (1,464), and per-field normalization statistics come from the training years only. The climatology reference used in evaluation is computed from training months. Forecast pairs never straddle a split boundary. A historical re-evaluation of the matched-step checkpoints reproduced their recorded values to within 3×10^{-8} and recovered the physical scale used in Fig. A3.

Land-cover data. BigEarthNet v2 (reBEN) with its official geographic split. The Sentinel-2 bands are resampled to the 10 m grid of the patch and Sentinel-1 VV and VH are used in decibels; the red band is the primary and the other thirteen channels are modifier tracks. The four-arm study uses a 14M-parameter model, a fixed 8,000-patch validation subsample for checkpoint selection, and a fixed 40,000-patch test subsample for scoring, so its absolute numbers are not comparable to full-test leaderboard entries and the earlier 85M screen. The four arms were not run under a common time budget: each trained with early stopping on validation macro-AP and was scored at its best validation checkpoint, which the summed arms reached after about 230k–255k steps and the factored arms after 125k–135k, and resumed runs reset their recorded clocks, so we report no per-arm training times. The per-channel models are slightly larger (14.0M versus 13.8M parameters) because their encoders are deeper. A metadata audit found no cross-split duplicate patch identifiers or footprints and no exact-pixel duplicates, and the official split keeps a substantial spatial buffer between training and test patches within a tile; validation patches are not buffered in the same way, acquisitions are shared across splits, and cross-tile proximity was not measured.

Earlier comparisons (Fig. A3). The matched-step WeatherBench comparison trained factored, hybrid, and summed models with per-channel encoders for 3,000 optimizer steps and three seeds. In physical units the test RMSEs are 158, 161, and 188 m^2/s^2; the hybrid used about 37.5% less wall

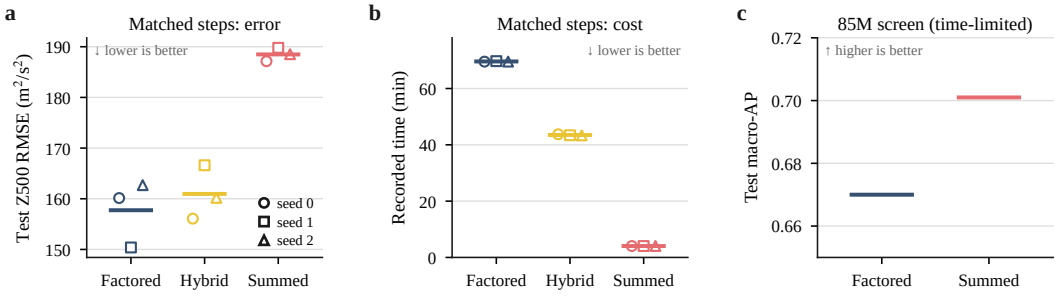


Figure A3: Earlier real-data comparisons, superseded by Fig. 4. (a) WeatherBench 2 test Z500 RMSE in m²/s² and (b) recorded training time after 3,000 matched optimizer steps with per-channel encoders, three seeds. (c) BigEarthNet-19 frozen-test macro-AP from the time-limited 85M-parameter screen (report-level means; per-seed values were not retained).

clock than factored, and summed far less than either. At equal steps the factored model receives more computation per step, which is why Fig. 4 replaces this protocol with equal time; the earlier result should be read as "at equal steps, more tokens help", not as a compute-matched advantage. The earlier BigEarthNet screen trained 85M-parameter factored and summed models with per-channel encoders until a 15-hour wall-clock limit, at which point neither had converged, and found 0.670 versus 0.701 macro-AP. Both earlier results are consistent with the per-channel condition of Fig. 4 and neither tests the shared-encoder condition.

F Reproducibility and Provenance

The figure pipeline records source hashes, builds statistical-unit tables and summaries from them, and regenerates every figure deterministically. The manuscript resolves its headline numbers from the same generated facts. The source contract distinguishes arm-level data from verified report-level summaries, and each figure carries a sidecar naming its inputs. Rendering a descriptive summary does not promote it to independently verified status. Throughout, *independently verified* means recomputed from arm-level records in a separate session, not replicated by an external group, and results not recomputed this way are marked as such.

Some historical limitations remain visible and we have left them so. The contact reach artifacts carry a documented hash-convention discrepancy, the placement runs lack some launch-time environment hashes, and local baseline runs were launched from working trees with uncommitted modifications. Later byte hashes and independent numerical checks support the records that exist but cannot recreate missing launch provenance. None of the core interpretation rests on the quarantined classification rung or on the failed-control baseline rankings. The mathematical claim is scoped to the accepted formal statement, whose archived package the manuscript restates without modification.