CHARTA: A RECURRENT LANGUAGE MODEL WITH ROUTED WRITES AND ADDRESSABLE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Fixed-state recurrent models provide constant-memory processing, but maintaining structured, persistent task-relevant state under memory load is not generally an explicit objective. We introduce CHARTA, an attention-free language model that structures delta-rule recurrent memory through routed and addressable writes. CHARTA retains the delta update while adding a block-level organiser that routes write strength across blocks, an event router that conditions write content and strength, and contextual and token-stable addressing paths. It can be trained with auxiliary supervision of task-relevant state while its recurrent matrices, which we call *boards*, remain accessible through fitted or learned readouts. We evaluate CHARTA along three complementary axes: persistence under load, state accessibility and causal relevance, and data-efficient language modelling. After training with at most four bindings, CHARTA reaches 96.7% accuracy with sixteen, versus 82.3% for DeltaNet, and retains once-stated facts across unseen window boundaries. A linear decoder recovers held-out item locations from boards at 49.0%, versus 26.1% for DeltaNet; board transplantation causally changes predictions. On BabyLM strict-small, CHARTA achieves a higher aggregate score than the top entry (55.9 vs. 53.6). Together, our results show that fixed-size recurrent state can be structured to retain, expose, and reuse task-relevant information without attention over the preceding context.

1 INTRODUCTION

Attention-based language models answer a query by returning directly to an earlier part of the visible context. A fixed-state recurrent model instead compresses the preceding sequence into memory of constant size. Because the state can be carried forward without growing with context length, fixed-state recurrence provides bounded-memory processing for streaming and long-context sequences. The corresponding difficulty is that information needed later must be preserved while new information continues to enter the same state. When a sequence states, changes, and later queries a fact, the model must decide what to retain, where to write it, and how a later query should retrieve it. A correct final answer does not show whether the relevant information remains in the recurrent state, whether that state controls the answer, or whether the same mechanism continues to work as more facts compete for limited capacity.

Existing fixed-state recurrent models provide increasingly sophisticated state updates, but typically do not make maintaining a structured, persistent, load-robust representation of task-relevant state a central design objective. Delta-rule memories correct the value stored at a selected address (Schlag et al., 2021; Yang et al., 2024b), and recent variants change when the state forgets or rewrites an association (Yang et al., 2025; Kimi Team, 2025; Hatamizadeh et al., 2026; Li et al., 2026; Siems et al., 2025; Goldstein et al., 2025; Hu et al., 2025; Park et al., 2026). These methods refine the recurrent update, but leave open how writes should be organised across the available state, which entities should return to stable addresses, and whether the resulting matrices can be read or changed directly. Evaluation also usually ends at model output, providing little evidence about what the recurrent state contains or whether it causally determines the prediction.

We address these questions through CHARTA, an attention-free recurrent language model with a structured fixed-size memory designed to organise how information is written and addressed, and to make the resulting state accessible to explicit readout and intervention. Each recurrent matrix

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.

is a *board*. A block-level organiser routes a token's write strength across blocks. An event router conditions the content and base strength proposed by each head. Contextual heads derive an address from the current representation, while object-file heads, whose addresses depend only on token identity, let repeated item words return to the same address. CHARTA retains the standard corrective delta rule; our novel contribution is how the update is formed, addressed, and distributed, together with direct access to the resulting boards. The boards can be supervised during training, read after training, and directly changed through intervention.

We evaluate three properties of CHARTA: persistence and generalisation under memory load, accessibility and causal relevance of task-relevant state, and data-efficient language modelling.

CHARTA improves over the matched recurrent baseline, DeltaNet (Yang et al., 2024b), where the recurrent state must hold several facts at once or expose them to a reader. Both recurrent models retain an isolated fact over long gaps, but they separate as memory load increases: after training with at most four person–code pairs, CHARTA reaches 96.7% accuracy with sixteen simultaneous pairs, compared with 82.3% for DeltaNet (three-seed means). Its boards also expose task-relevant state: a frozen linear decoder recovers held-out item locations substantially better than matched recurrent controls, and board transplantation (replacing one run's boards with another's) causally changes the model's answer. On BabyLM strict-small, CHARTA remains competitive as a language model, while reaching a higher aggregate score than the top leaderboard entry.

Our contributions are three-fold: 1) We introduce CHARTA, a recurrent language model that structures fixed-size delta-rule memory through routed and addressable writes, keeping the corrective delta update while adding block-level routing of write strength, event-conditioned write content and strength, and complementary contextual and token-stable addressing; 2) We show that this organisation improves recurrent memory under concurrent load, separating retention of an isolated fact from maintaining multiple associations in the same fixed-size state, and that CHARTA generalises beyond its training load as the number of simultaneous bindings grows, with ablations identifying write routing as the most important component under high load; 3) We make the carried recurrent matrices directly testable as memory states in three ways: through auxiliary supervision during training, fitted readouts from frozen boards, and board transplantation, which together test what the memory contains and whether it controls the answer.

2 Related Work

Recurrent matrix memories. Linear attention keeps a matrix that it updates once per token. Fast-weight programming connected this view to a delta update that corrects the value already stored at an address (Schlag et al., 2021), and DeltaNet made this update parallel over sequence length (Yang et al., 2024b). Subsequent recurrent and linear-attention architectures, including Mamba, GLA, Gated DeltaNet, Kimi Delta Attention (KDA), and DeltaProduct, introduce gating, forgetting, erasure, or other state transitions (Yang et al., 2024a; Gu & Dao, 2023; Fu et al., 2023; Yang et al., 2025; Kimi Team, 2025; Hatamizadeh et al., 2026; Li et al., 2026; Siems et al., 2025; Goldstein et al., 2025; Hu et al., 2025; Park et al., 2026). These updates primarily decide when the state forgets or rewrites an association. They leave open how writes should be distributed across blocks, whether repeated entities should return to stable addresses, and whether the resulting matrices can be directly read or edited. CHARTA addresses each of these explicitly.

Routed and supplementary memory. Mixture-of-Memories routes tokens among several recurrent memories, while Sparse Delta Memory increases capacity through sparse reads and writes to a larger store (Du et al., 2025; Cabannes et al., 2026). Infini-attention, Titans, and HOLA place a memory or exact cache beside another sequence mechanism (Munkhdalai et al., 2024; Behrouz et al., 2025; Cui, 2026). MeMo directly constructs correlation matrices that store token sequences (Zanzotto et al., 2025). These approaches either add a store beside the recurrent state or route among multiple stores. In both cases the carried matrices are written without coordination across blocks and are not themselves read or intervened on, whereas CHARTA coordinates each token's write across its boards and reads and transplants them directly.

Entity representations. Entity Networks and EntityNLM maintain representations indexed by entities (Henaff et al., 2017; Ji et al., 2017). Object-file theory similarly describes persistent rep-

resentations that follow an object as its properties change (Kahneman et al., 1992). These models were built for entity-centred tasks rather than as general language models, and do not combine stable entity addressing with a shared delta-rule matrix memory. CHARTA's object-file heads bring this principle into a delta-rule language model.

Supervising current state. Breakpoint Transformers train intermediate beliefs that can be queried at marked points in a sequence (Richardson et al., 2022). Situation Supervision teaches language models to encode entity-state variables in their hidden representations (Li et al., 2023). These studies establish direct supervision of internal state as a useful training signal, but do not compare how different recurrent architectures respond to the same signal. CHARTA applies the same training-only state-prediction head to matched recurrent and attention controls and separately measures what can be recovered from its boards, which separates the effect of the teaching signal from the architecture.

Reading and changing internal state. Linear decoders recover game state and propositions from model activations (Nanda et al., 2023; Feng et al., 2025), while interventions on recurrent networks identify mutable internal variables (Grant et al., 2025). These studies read or edit activations in attention models or trained recurrent networks; however, they leave open whether the same operations apply to fixed-size recurrent matrices rewritten at every token. Weight-editing methods such as ROME instead alter stored model knowledge by changing model parameters (Meng et al., 2022). CHARTA keeps the weights fixed and reads and transplants its carried boards, testing whether changing them changes the answer without disturbing unrelated facts. We generate tracking, deduction, and binding tasks in the controlled diagnostic style of bAbI (Weston et al., 2015).

3 CHARTA: Structured Recurrent Memory

CHARTA is a standard pre-norm decoder in which attention is replaced by the recurrent mixer shown in Figure 1, which traces the data flow from input IDs to next-token probabilities. The mixer maintains persistent delta-rule boards and structures how writes are formed, addressed, and distributed across blocks.

Terminology used. A *token* is one input symbol and h_t is the vector representing it at the current block. A *board* is one head's recurrent matrix M; *recurrent memory* denotes the collection of boards across all blocks. *Stories* are short synthetic texts, such as "Box 1 contains the cake", on which the controlled tasks are built. An *entity* is a person or object in a story and may be mentioned by several tokens. Each head derives a write address k_t, a content vector v_t, a read cue q_t, and a write strength β_t. The address selects what part of the board is changed, the content vector what that address should return, and the read cue what is retrieved, while the write strength, between zero and one, sets how much of the proposed correction is applied. These correspond to the key, value, query, and gate in linear attention. All linear maps include a bias; we write only those needed in the equations. The write address is L2-normalised before the update. Its length cannot amplify a write, since address matching depends on its direction.

Event-conditioned writes. Let $\bar{h}_t^{(\ell)}$ and $\tilde{h}_t^{(\ell)}$ denote the normalised mixer input and post-mixer residual in block ℓ. Linear projections followed by a short causal depthwise convolution produce the contextual addresses, read cues, and content. The width-4 convolution lets a write depend on the current token and its local left context without exposing future tokens. Contextual heads use local word order before writing to long-term memory, while the object-file address and cue path bypasses the convolution to preserve exact token identity. In a matched language-model ablation, the convolution lowers perplexity by 0.30. The event router gives the token eight scores, and softmax turns them into a mixture of eight learned write channels:

$$p_t^{(\ell)} = \mathrm{softmax}(W_e \bar{h}_t^{(\ell)} + b_e). \tag{1}$$

The same mixture $p_t^{(\ell)}$ conditions both the proposed content and the base write strength: one learned table adjusts $v_t^{(\ell)}$, and another adjusts $b_t^{(\ell)}$. With $\mathcal{C}_4$ denoting the width-4 causal convolution:

$$v_t^{(\ell)} = \tanh\Big(\mathcal{C}_4(W_v \bar{h}_t^{(\ell)})\Big) + p_t^{(\ell)} B_v, \tag{2}$$

$$b_t^{(\ell)} = \alpha\, \sigma(W_\beta \bar{h}_t^{(\ell)} + p_t^{(\ell)} B_\beta). \tag{3}$$

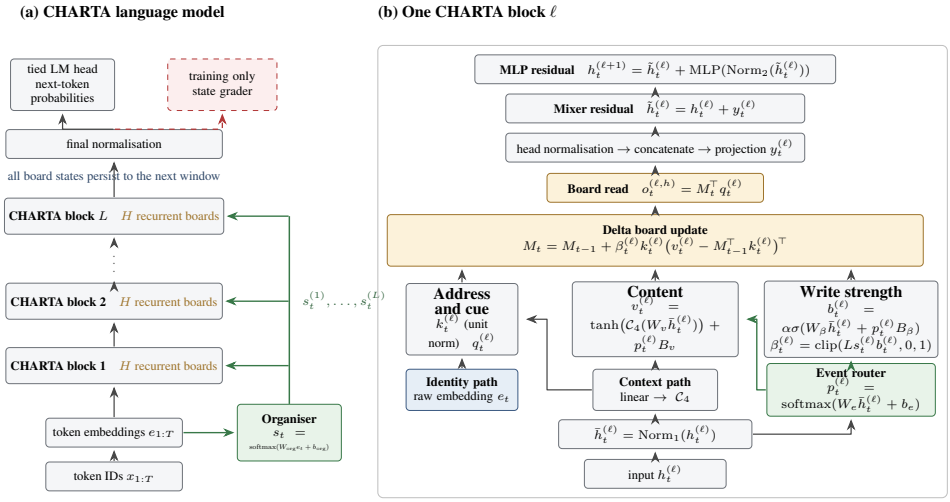

Figure 1: CHARTA data flow ($L = 6$). **(a)** The organiser sets how strongly each token writes to every block's boards, which persist between windows. The dashed grader is training-only. **(b)** Each head uses an identity-based or contextual address, and the event router adjusts its content and write strength.

α is the fixed global write scale. We bound the base value target with $\tanh$ before adding the event-dependent correction, so that a value written at token 10 stays comparable with one written at token 500 and later writes cannot dominate earlier associations through scale alone (Section 6 tests the resulting training stability). The eight channels receive no labels such as "move" or "give". They are learned choices for adjusting content and write strength under the same recurrent update.

Routing write strength across blocks. For each token, the organiser reads the raw embedding e_t and produces one share for every block:

$$s_t = \mathrm{softmax}(W_{\mathrm{org}} e_t + b_{\mathrm{org}}) \in \mathbb{R}^L. \tag{4}$$

The shares sum to one. Each head computes its own base strength $b_t^{(\ell)}$, with the head index suppressed. The organiser does not choose a head's address or content; it only redistributes write strength across blocks. Specifically, $L s_t^{(\ell)}$ multiplies every head's base strength in block ℓ. The final write strength is $\beta_t^{(\ell)} = \mathrm{clip}(L s_t^{(\ell)} b_t^{(\ell)}, 0, 1)$. Under uniform routing, $s_t^{(\ell)} = 1/L$, so the multiplier is one and every block retains its original base strength. The organiser is zero-initialised at this neutral condition. Training can then make the shares unequal: a larger share strengthens writes in that block, while a smaller share weakens them. Clipping to $[0, 1]$ prevents the organiser from pushing the update beyond the proposed content.

Delta-rule board update and read. Each head maintains $M_t \in \mathbb{R}^{d_k \times d_v}$. Before writing, the head retrieves the value currently associated with k_t. The difference between this retrieved value and the proposed content v_t becomes the correction applied to the board. The updated board is then read using q_t:

$$M_t = M_{t-1} + \beta_t k_t \big(v_t - M_{t-1}^\top k_t\big)^\top, \tag{5}$$

$$o_t = M_t^\top q_t. \tag{6}$$

This is the standard delta rule (Yang et al., 2024b), and CHARTA does not alter the recurrence itself. The subtraction first recovers the value currently returned at k_t and writes only the error relative to v_t; e.g., if a story changes the holder of a key from Alice to Bob, the delta update corrects the value returned at the key's current-holder address toward Bob instead of superposing Alice and Bob at that read. A matched binding comparison rises from 9% with additive writing to 71% with corrective writing, which is why CHARTA retains the delta rule. After a shared per-head normalisation, head outputs are joined, projected, and returned through the mixer and MLP residuals shown in Figure 1.

Stable addresses for repeated entities. Contextual heads derive k_t and q_t from $\bar{h}_t^{(\ell)}$. A fixed subset instead applies the same address and cue projections to the raw token embedding e_t, without the convolution, so the same item word returns to one address across sentences and positions. Their content v_t still comes from the contextual path. We call these *object-file heads*, after the account of stable object representations in Kahneman et al. (1992), because their addressing is token-stable rather than contextual. CHARTA therefore combines token-stable addressing with contextual content within the same delta-rule memory. The default configuration uses two object-file heads among six. A stable address supports one-shot binding and persistence, while a purely contextual address relocates a moved fact more readily. Section 6 quantifies both sides of this trade-off and motivates why the heads remain in the default configuration.

Causal order and streaming. CHARTA uses no learned positional embedding. Order enters through the causal convolution and sequential board updates. At a window boundary, every board is carried to the next window exactly, while the convolution is reset to match the windowed semantics used during training. The model can process an arbitrary number of windows without increasing the size of its recurrent memory.

Training-only supervision of task-relevant state. After a sentence changes a holder, location, or ordering relation, a small linear state-prediction head (*grader*) reads the final normalised hidden vector at that position. It adds an auxiliary loss for predicting all currently correct facts, such as the current holder or each person's position in an ordering. We grade after these updates because the complete set of current facts is defined there; positions inside a sentence may not yet have a completed update to grade. The grader supervises hidden representations rather than the boards themselves and is not part of the model's word-prediction path.

4 Experimental Setup

Tasks and measurements. We use BabyLM strict-small, controlled synthetic tasks, and the MAD architecture benchmark (Poli et al., 2024). BabyLM strict-small (Warstadt et al., 2025; Timiryasov & Tastet, 2023) evaluates language models trained on limited data. Its zero-shot evaluations comprise BLiMP and its supplement (choosing the grammatical sentence in a minimal pair (Warstadt et al., 2020)), EWoK (basic world knowledge (Ivanova et al., 2025)), COMPS (property inheritance between concepts (Misra et al., 2023)), GlobalPIQA (culturally varied physical-commonsense questions (Chang et al., 2025)), and Entity Tracking (who holds or contains an item as a story changes (Kim & Schuster, 2023)). A fine-tuned (Super)GLUE aggregate over seven classification tasks (Table 8) forms the seventh evaluation column. MAD, the mechanistic architecture design benchmark (Poli et al., 2024), places each candidate mixer in the same two-block setup and evaluates six synthetic recall, compression, memorisation, and copying tasks over 66 settings (Appendix I defines these tasks). Finally, we generate controlled synthetic datasets, whose generation scripts we will release with the source code, each described with its results in Section 5 and illustrated in Appendix N. Several use *box stories*, short texts stating which item each numbered box contains, sometimes followed by moves such as "Move the bell from Box 3 to Box 2", after which the model is asked what a box contains. Box-location decoding fits on 2,000 stories, selects on 500, and tests on 1,000, and transitive deduction trains on 12,000 and tests on 1,500. Person–code and long-gap recall examples are generated fresh during training and evaluated on 200 examples per condition, and the relocation and transplantation tests use 200 and 1,000 stories.

Experimental configurations and baselines. The BabyLM configuration has six blocks, width 384, six recurrent heads, a 256-token training window, and no positional embeddings, and is trained within the 10M-word limit. The controlled synthetic experiments use compact task-specific configurations to isolate individual memory properties (Appendix H). We use matched DeltaNet as the primary recurrent baseline wherever a direct mixer replacement is possible, since CHARTA retains the same corrective delta-rule recurrence. The transitive-deduction and person–code binding experiments additionally include Transformer++, a transformer with rotary position embeddings (RoPE), SwiGLU, and RMSNorm (Arora et al., 2024; Su et al., 2024). Board decoding additionally includes Kimi Delta Attention (KDA) and Gated DeltaNet; long-gap recall includes a windowed RoPE transformer as an attention control; and MAD uses the official baselines. The DeltaNet replacement uses the authors' published flash-linear-attention (FLA) layer (Yang & Zhang, 2024) under the same

training setup, corpus, schedule, and approximate parameter range. For board decoding, DeltaNet, KDA, and Gated DeltaNet replace every CHARTA mixer under the same corpus, training command, evaluation, and seed; model sizes are within 6%. Comparisons reported as mean and sample standard deviation use three seeds. The remaining comparisons use one matched seed. Appendix T gives the complete replication scope.

5 RESULTS

5.1 PERSISTENT MEMORY AND GENERALISATION UNDER LOAD

We first test whether recurrent state preserves an isolated fact after its source words leave the model's input window. In the controlled synthetic long-gap recall task (Appendix N), the model is given a person paired with one of sixteen arbitrary nonsense codes (e.g., "Alice secret word zorp"), followed by D tokens of unrelated natural language text, and is then asked to recover that person's code. The intervening text contains no information about the answer, so success requires retaining the original fact. CHARTA and the matched DeltaNet are trained only with $D \leq 192$, while evaluation extends to gaps of 768 tokens. Both models carry their complete recurrent states across successive 256-token windows and recover the fact at 100% throughout, whereas a windowed attention control, which receives only the current window, falls to chance once the source sentence is no longer visible (Appendix A). This shared result shows that persistence of an isolated fact follows from carried recurrent state.

We next increase the memory load by requiring several person–code facts to remain available at once, training both CHARTA and DeltaNet with at most four pairs and evaluating with up to sixteen. At a fixed 512-token gap, CHARTA scores 100/100/100/99.8/94.2% with 1/2/4/8/16 pairs, compared with 100/100/99.8/97.3/82.0% for DeltaNet (Appendix A). The in-window evaluation in Section 5.2, where all sixteen pairs remain visible, shows the same pattern. Both models thus preserve a single fact over long gaps, but CHARTA retains more simultaneous facts as the load exceeds the training regime. Here CHARTA writes twice per token (Siems et al., 2025), and Appendix A tests one write.

5.2 TEACHING COMPLETE TASK STATE: ORDER AND PERSON–CODE BINDINGS

We next ask whether auxiliary supervision can encourage the model to retain the complete task-relevant state, rather than only the information needed to answer the current query. We test this with two controlled synthetic tasks: transitive deduction and person–code binding (Appendix N). In the transitive deduction task, a complete ordering of eight people is sampled and the model is shown shuffled comparisons between adjacent people in that order. For example, after "Alice is taller than Bob; Bob is taller than Clara", the query "Is Alice taller than Clara?" requires the language model to answer *yes*, not the complete order. Queries span chains of one to seven comparisons, so longer queries require composing several stated relations. The standard next-token prediction loss supervises the requested yes/no answer, but does not require the final hidden representation to make every person's position in the underlying order recoverable.

During training, we therefore add a linear grader that reads the hidden vector after the comparisons. For each of the eight people, it predicts one of eight positions in the complete order (12.5% chance), and its prediction loss is added as auxiliary supervision. This explicitly supervises the complete ordering in addition to the language-model answer. The grader never supplies the model's yes/no prediction. After training, we freeze the same grader and evaluate it on held-out deduction stories to measure how much of the complete order the final hidden vector retains.

Person–code binding follows the same design. Given "Alice secret word zorp; Bob secret word blick" and "Bob secret word ...", the language model target is *blick*, while the auxiliary grader predicts the code belonging to every stated person. Training contains at most four simultaneous pairs, while the hardest test contains sixteen, all inside the input window. The language model selects the queried code from sixteen candidates (6.25% chance).

On transitive deduction, CHARTA's yes/no language-model answers reach $91.3 \pm 7.0\%$ on held-out stories, compared with $78.4 \pm 0.8\%$ for matched DeltaNet, while Transformer++ remains near chance at 52.3% (Table 1). The frozen auxiliary grader predicts each person's position from CHARTA's final hidden vector at $98.2 \pm 1.1\%$, compared with $38.1 \pm 0.4\%$ for DeltaNet.

Task	Chance	CHARTA	DeltaNet	T++
Yes/no deduction	50.0	91.3±7.0	78.4±0.8	52.3
Position (8-way)	12.5	98.2±1.1	38.1±0.4	–
4 person–code pairs	6.25	100±0	100±0	27.0
16 person–code pairs	6.25	96.7±4.2	82.3±3.6	6.5

Table 1: Teaching comparison (%). *Position* is the frozen grader, other rows are model answers, ± is SD over three seeds.

Model	Box loc.	7-task	MultiRC	F1	Zero-shot
CHARTA	**49.04**	**+1.47**	**62.13**	**0.49**	54.44
DeltaNet	26.08	+0.23	57.55	0.000	–
KDA	21.83	–	57.67	0.008	**54.61**
Gated DeltaNet	20.56	–	57.55–58.21	0.000–0.217	52.64

Table 2: Information read from boards. *Box loc.* decodes unseen items (chance 20), and *7-task* is the gain from board reads. CHARTA's 7-task and MultiRC are three-seed means, Zero-shot is one run.

On person–code binding (Table 1), both recurrent models answer perfectly with four pairs. At sixteen pairs, CHARTA reaches 96.7%, DeltaNet 82.3%, and Transformer++ 6.5%.

We also train a single CHARTA model on both long-gap recall and person–code recall tasks at once, using a 50/50 mixture of ordinary text and automatically generated recall examples (Appendix B). It retains 98/95/86% long-gap recall at gaps of 320/512/768 tokens and 94–96% person–code recall with 1–16 pairs. Compared with a pure-text CHARTA control, its perplexity is higher at matched compute because recall examples displace text, but only by 0.03 when both see the same amount of language-modelling data.

5.3 Reading facts from boards and causal transplantation

Section 5.2 reads the final hidden representation. We now examine the boards, asking whether task-relevant information can be recovered from them and whether changing them changes the prediction.

Linear readout from the frozen boards. We use the synthetic box stories introduced in Section 4, in which four items are placed in five boxes and up to three items are moved. After pretraining, every model is frozen and processes the stories. We then train a linear decoder to identify the current box of a queried item using only the final boards (with five candidate boxes, chance accuracy is 20%). For candidate box b and item word w, the decoder scores the feature $[(M^\top q_b) \odot v_w, (M^\top k_b) \odot v_w]$, which is linear in the board M. Here q_b and k_b are the model's own read cue and write address for box b, and v_w represents the item. Item words are disjoint between decoder training and testing, so the decoder must locate items it never saw during training. All architectures share the decoder setup, item split, and three classifier seeds (Appendix S). The decoder recovers the correct box for 49.04% of held-out items from CHARTA's boards, compared with 26.08% for matched DeltaNet, 21.83% for KDA, and 20.56% for Gated DeltaNet (Table 2). KDA is the control closest to CHARTA in language model performance, with a zero-shot BabyLM mean of 54.61 against 54.44 (Table 2) and held-out perplexity of 32.89 against 33.53 (Appendix I), yet its boards decode near chance. Similar language-model quality therefore does not imply similarly accessible recurrent state.

We next test whether the boards can improve downstream predictions. On the seven BabyLM (Super)GLUE tasks, we compare the standard fine-tuning setup, in which the classifier uses only the final hidden vector, with an otherwise identical classifier that also receives eight learned reads from the final boards, added to the final hidden vector through a zero-initialised projection so that both classifiers start identical. This raises CHARTA's seven-task mean by 1.47 ± 0.31 points across three seeds, against $+0.23$ for DeltaNet (Table 8, Appendix M). On MultiRC, where predicting only the negative class gives 57.55 accuracy with zero F1, CHARTA's board-assisted classifier reaches 62.13 with $\mathrm{F1} = 0.489$, while every other tested classifier input has F1 at or below 0.217 (Table 2).

Causal transplantation. Here, we test whether the boards causally influence CHARTA's answer rather than merely contain it. We process two box stories that differ only in the item in one box, for example the bell in Box 2 in the recipient story and the coin in the donor story. Before the recipient story is queried, its boards are replaced with the donor's and CHARTA scores *bell* and *coin* as continuations of "Box 2 contains the" (Appendix N). Across 1,000 stories, transplanting all boards makes CHARTA prefer the donor's item on 99.8% of cases. The contextual heads' boards alone carry the change (99.9%, against 0.4% for the object-file heads), so the controls transplant those. A donor holding a deliberately incorrect item makes CHARTA prefer that item on 99.8% of cases, an unchanged box keeps its answer on 99.7%, and a donor in which only a different box

changed alters the target answer on 0.0%. A DeltaNet-equivalent control built in CHARTA's code, without object-file heads, organiser, or learned event routing, is also editable (93.9%, with a 5.0% change from different-box donors), but its boards expose less information to the linear decoder.

5.4 Data-efficient language modelling

Structuring the recurrent memory should not come at the cost of ordinary language modelling. We test this on BabyLM strict-small, where models are trained on at most 10M words and evaluated with the official zero-shot suite and fine-tuned (Super)GLUE tasks (Section 4). CHARTA is trained for ten passes over this budget, which includes 1.0M words of automatically generated tracking stories (Appendix H). We compare it with the top entry of the archived leaderboard (Table 3).

CHARTA scores more than twice the archived entry on entity tracking (60.76 against 27.95) and is also higher on GlobalPIQA. On every metric where it scores lower, it reaches at least 91% of the entry's score. Its zero-shot mean is 54.44 against 50.78, and its seven-column average is 55.85 against 53.59. Table 3 reports one CHARTA run, and three pretraining seeds give a zero-shot mean of 53.95 ± 0.43 and entity tracking of 59.97 ± 0.84. The entity-tracking difference cannot be attributed to the architecture alone, because the training mixture includes the tracking stories above, and matched DeltaNet trained on it reaches a similar score (60.21). When entity training data is scarce, however, CHARTA reaches 44.65, compared with 34.41 for DeltaNet and 16.19 for matched attention (Appendix L). On (Super)GLUE, CHARTA scores lower (64.33). Encoder and hybrid models lead this column, topped by the archived entry itself at 70.44, and the nearest decoder-only entry reaches 66.64. CHARTA's score already uses the board-assisted classifier from Section 5.3. Within the visible window, a matched transformer has lower perplexity at every measured data size, consistent with attention keeping direct access to recent tokens, although the relative gap falls from 48% at 0.3M words to the mid-teens from 7M words (Appendix G). Because CHARTA's memory is fixed in size, attention decodes faster only up to 4k tokens, and CHARTA is faster at 16k.

Matched mixer comparison on MAD. The BabyLM leaderboard comparison changes data, objective, and architecture at once. We therefore use MAD to compare the sequence mixers under a common experimental setup. Each candidate replaces the mixer in the same two-block model and is evaluated on six synthetic tasks over 66 settings, taking the best of the same six optimiser choices in each setting (Appendix I). We run CHARTA in this setup without modification and with one configuration across all tasks. Its average of 73.2 is second to Transformer++ (74.5) and above Mamba (69.3), GLA (60.0), and DeltaNet (71.8) (Table 4). CHARTA's largest MAD advantage is on fuzzy recall, where each key and value is a group of one to three adjacent tokens: CHARTA scores 43.7 versus 35.7 for DeltaNet and 29.8 for Transformer++ (Figure 5, Appendix I). Because CHARTA's main configuration uses wider addresses, we double address width in both models, which raises CHARTA from 30.6 to 42.0 but lowers DeltaNet from 29.6 to 26.8, so address width alone does not explain the difference (Appendix Q).

6 Ablations and Analyses

Each ablation removes one CHARTA component, retrains under the same final setting, and is evaluated on the full BabyLM suite and on the synthetic tests targeting that component. Appendix K reports all eight combinations of the following three components.

Organiser. The organiser redistributes write strength across blocks, so we test it as the number of stored facts grows. Removing it lowers BabyLM entity tracking by 7.9 points. On in-window person–code binding (Section 5.2), adding the organiser to an otherwise identical model raises accuracy with sixteen pairs from 69.5% to 92.0%, while accuracy with four pairs stays at 100%.

Object-file heads. Object-file heads give a repeated item word the same address (Section 3). In a box story traced sentence by sentence (Figure 3), the model is asked what every box contains. After a fact is first stated, the full model places 0.74 probability on the correct content. Removing only the object-file heads lowers this to 0.37, matching DeltaNet, and the fact is retained reliably only after repetition. The same address stability creates a trade-off when facts move. In the relocation test, an item moves to an empty box and the model is asked what that box now contains. The full

	BLiMP	Supp.	EWoK	Entity	COMPS	GlobalPIQA	Zero-shot	(Super)GLUE	7-col. avg
CHARTA	66.31	58.26	49.78	**60.76**	51.89	**39.64**	**54.44**	64.33	**55.85**
Archived entry	**71.60**	**63.93**	**51.94**	27.95	**53.13**	36.15	50.78	**70.44**	53.59

Table 3: BabyLM strict-small. CHARTA is one reported run (three-seed result in the text), compared with the top entry of the archived 29 July 2026 leaderboard.

Model	Compress	Fuzzy	ICR	Memorize	Noisy	SelCopy	Avg
Transformer++	51.6	29.8	94.1	85.2	86.8	99.6	**74.5**
Mamba	52.7	6.7	90.4	**89.5**	90.1	86.3	69.3
GLA	38.8	6.9	80.8	63.3	81.6	88.6	60.0
DeltaNet	42.2	35.7	100	52.8	100	100	71.8
CHARTA (fixed setting across tasks)	40.6	**43.7**	99.9	55.1	99.9	99.8	**73.2**

Table 4: MAD comparison with published baselines (Appendix I). CHARTA uses one setting for all tasks (73.8 if chosen per task). ICR and SelCopy are in-context recall and selective copying.

model selects the moved item on 0.5% of stories; removing the object-file heads raises this to 81%, while DeltaNet reaches 42%. The full model still answers in-place replacements, which overwrite an existing address, at 99.7% (Appendix S.1). Although relocation accuracy is low, a linear decoder still recovers the updated fact, so the update is stored but not selected, a read-arbitration problem rather than a failed write (Appendix D). On the BabyLM tasks, the full and no-object-file configurations are similar (zero-shot means 54.44 and 54.33), with entity tracking intact (Appendix K). Object-file heads therefore trade easier one-shot binding and persistence for slower relocation, making the choice task-dependent. We retain them by default because the decoding, transplantation, and person–code experiments require facts to be stated once and persist; settings with frequent revision may instead favour removing them. The no-object-file configuration is a single seed.

Bounded content. CHARTA passes the base content through $\tanh$ before the delta update, so later writes cannot dominate earlier ones through scale alone (Section 3). We test whether this bounding improves training stability on the hardest MAD memorisation setting, where the unbounded DeltaNet run reaches 5.6% accuracy with perplexity 347. CHARTA with only the bounded write reaches 75.5%, and variants adding the other components reach 85.9–97.9%. DeltaNet obtains a strong result under its authors' larger sweep, so this comparison concerns our matched settings.

Event router. The event router mixes eight learned write channels that adjust the content and strength of each write. On MAD fuzzy recall with vocabulary 32 and the organiser present, removing it lowers accuracy from 37.9% to 12.9%. On BabyLM, where entity training examples are abundant, removing it does not clearly change the mean but increases seed variation. In the full component-combination comparison (Appendix K), the router raises the BabyLM mean with the organiser but slightly lowers it with only the object-file heads. Its benefit therefore depends on the organiser and on how scarce training data is, and is clearest under associative interference as in fuzzy recall.

7 Conclusion

CHARTA structures a fixed-size delta-rule memory through routed and addressable writes. In long-gap recall, CHARTA and matched DeltaNet both retain a single fact after it leaves the input window, but CHARTA retains more simultaneous associations as load exceeds training levels. Task-relevant state is also accessible and causally relevant: auxiliary supervision makes a complete eight-person order recoverable from CHARTA's final hidden vector, a frozen linear decoder recovers unseen item locations from its boards substantially better than from DeltaNet, KDA, or Gated DeltaNet, and transplanting its boards changes the prediction to the donor's value. As a language model, CHARTA exceeds the top archived BabyLM strict-small entry on the seven-column average and remains competitive in the matched MAD comparison, while a matched transformer has lower in-window perplexity. Together, these results show that fixed-size recurrent memory can be structured to retain task-relevant information under load and make it recoverable and causally effective. Scaling CHARTA and improving read arbitration remain future work; Appendix T states the limitations.

Ethics statement

This work uses publicly available datasets (the BabyLM strict-small data and TinyStories) and synthetic stories generated by our own scripts. It doesn't involve any human subjects, personal data or crowdsourced annotation.

AI use statement

The research questions, model architecture, experimental design, interpretation of results, and conclusions were developed by the authors. LLM-based tools were used as coding and writing aids. They assisted with implementation of the model, training, and evaluation code based on the authors' existing code and designs, and were also used to discuss experiments and results. For writing, LLMs assisted with manuscript editing, literature search, figure preparation, and reference formatting. All AI-assisted outputs were reviewed and verified by the authors, and every reported result was reproduced from evaluation runs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility statement

Section 4 presents the training and evaluation protocol, seeds, and replication for each comparison. Appendix H lists hyperparameters, Appendix I specifies the matched baseline architectures, Appendix K reports every component-ablation configuration, and Appendix N gives worked examples of each evaluation task. The source code, including the scripts that generate the synthetic data, will be released upon publication.

References

Simran Arora et al. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2312.04927.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html. arXiv:2501.00663.

Nghia Tuan Bui, Guergana K. Savova, and Lijing Wang. Assessing the macro and micro effects of random seeds on fine-tuning large language models. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 41–46. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-short.3. URL https://aclanthology.org/2025.ijcnlp-short.3/. arXiv:2503.07329.

Loïc Cabannes, Pierre-Emmanuel Mazaré, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Ilze Amanda Auzina, Justin Carpentier, Gabriel Synnaeve, and Hervé Jégou. Sparse delta memory: Scaling the state of linear RNNs through sparsity. *arXiv preprint arXiv:2607.07386*, 2026.

Tyler A. Chang, Catherine Arnett, Abdelrahman Sadallah, Abdelrahman Eldesokey, et al. Global PIQA: Evaluating commonsense reasoning across 100+ languages and cultures. *arXiv preprint arXiv:2510.24081*, 2025.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets. *arXiv preprint arXiv:2607.02303*, 2026.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*, 2025.

Jiahai Feng, Stuart Russell, and Jacob Steinhardt. Monitoring latent world states in language models with propositional probes. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/

hash/3132d0405fabe24b2a7b6cd7ba9de6b5-Abstract-Conference.html. arXiv:2406.19501.

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. arXiv:2212.14052.

Daniel Goldstein, Ruichong Zhang, Bo Peng, et al. RWKV-7 "goose" with expressive dynamic state evolution. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=ayB1PACN5j. arXiv:2503.14456.

Satchel Grant, Noah D. Goodman, and James L. McClelland. Emergent symbol-like number variables in artificial neural networks. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=YPnYpiru5W. arXiv:2501.06141.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1612.03969.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Comba: Improving bilinear RNNs with closed-loop control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.02475.

Anna A. Ivanova, Aalok Sathe, Benjamin Lipkin, et al. Elements of world knowledge (EWoK): A cognition-inspired framework for evaluating basic world knowledge in language models. *Transactions of the Association for Computational Linguistics*, 13:1245–1270, 2025. doi: 10.1162/tacl.a.38. URL https://aclanthology.org/2025.tacl-1.57/. arXiv:2405.09605.

Yangfeng Ji et al. Dynamic entity representations in neural language models. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017. arXiv:1708.00781.

Daniel Kahneman, Anne Treisman, and Brian J. Gibbs. The reviewing of object files: Object-specific integration of information. *Cognitive Psychology*, 24(2):175–219, 1992. doi: 10.1016/0010-0285(92)90007-O. URL https://doi.org/10.1016/0010-0285(92)90007-O.

Najoung Kim and Sebastian Schuster. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.acl-long.213/.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. Language modeling with latent situations. In *Findings of the Association for Computational Linguistics: ACL*, pp. 12556–12571. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.795. URL https://aclanthology.org/2023.findings-acl.795/.

Xiang Li, Chong Zhang, Hao Luo, et al. Erase-then-delta attention: Decoupling erase and write addresses in delta-rule linear attention. *arXiv preprint arXiv:2606.26560*, 2026.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2202.05262.

Kanishka Misra, Julia Taylor Rayz, and Allyson Ettinger. COMPS: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. In *European Chapter of the Association for Computational Linguistics (EACL)*, 2023. arXiv:2210.01963.

Tsendsuren Munkhdalai et al. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

Sumin Park, Seojin Kim, and Noseong Park. Q-Delta: Beyond key-value associative state evolution. *arXiv preprint arXiv:2606.08804*, 2026. URL https://arxiv.org/abs/2606.08804.

Michael Poli, Armin W. Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 40908–40950. PMLR, 2024. URL https://proceedings.mlr.press/v235/poli24a.html. arXiv:2403.17844.

Nils Reimers and Iryna Gurevych. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*, 2018.

Kyle Richardson, Ronen Tamari, Oren Sultan, Dafna Shahaf, Reut Tsarfaty, and Ashish Sabharwal. Breakpoint transformers for modeling and tracking intermediate beliefs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9703–9719, 2022. doi: 10.18653/v1/2022.emnlp-main.658. URL https://aclanthology.org/2022.emnlp-main.658/.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via householder products. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.10297.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Inar Timiryasov and Jean-Loup Tastet. Baby llama: Knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. In *Proceedings of the BabyLM Challenge at CoNLL*, 2023. arXiv:2308.02019.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. arXiv:1912.00582.

Alex Warstadt, Aaron Mueller, Leshem Choshen, et al. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. *arXiv preprint arXiv:2504.08165*, 2025.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015. URL https://arxiv.org/abs/1502.05698.

Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention mechanisms. https://github.com/fla-org/flash-linear-attention, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 2024a. URL https://proceedings.mlr.press/v235/yang24ab.html. arXiv:2312.06635.

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)*, 2024b. arXiv:2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.06464.

Fabio Massimo Zanzotto, Elena Sofia Ruzzetti, Giancarlo A. Xompero, Leonardo Ranaldi, Davide Venditti, Federico Ranaldi, Cristina Giannone, Andrea Favalli, and Raniero Romagnoli. Position paper: MeMo: Towards language models with associative memory mechanisms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15169–15180. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.785. URL https://aclanthology.org/2025.findings-acl.785/. arXiv:2502.12851.

A Recall After the Original Sentence Leaves the Window

A fixed-size recurrent memory must preserve information when processing crosses a window boundary. We test this property after removing surface clues: a fact is stated once, followed by D tokens of unrelated text and a query about the original fact. An ordinary fill-in-the-blank story does not isolate memory because a windowed transformer reaches 53.2% from visible characters, pronouns, and discourse markers. These cues are removed from the reported test.

Figure 2 gives the result. CHARTA and DeltaNet carry their complete recurrent states between 256-token windows; the RoPE transformer (Su et al., 2024) receives only the final window. All three models answer correctly while the fact remains in view. Once it leaves, the transformer falls to its measured chance rate of 6.0–8.0%, while CHARTA and DeltaNet remain at 100% through a gap of 768 tokens. Neither recurrent model trained on examples that crossed a window boundary. This establishes persistent recall as a benefit of carried recurrent state; the comparison between CHARTA and DeltaNet appears when that state must hold several facts at once.

Natural-language recall. Replacing the nonsense codes with ordinary sentences ("Lily's favourite colour was red.") across four attribute families, followed by a fill-in-the-blank question with eight plausible values, CHARTA again retains 100% at every gap through 768 while the windowed control sits at chance (10.5–13.5%, chance 12.5%). With four such facts stored at once, CHARTA answers at 92.5% from beyond the window, against 70% for the transformer with every fact still in view.

Recall with multiple simultaneous facts. Table 5 holds 1–16 person–code pairs at once, averaged over three seeds. Both models trained with at most four. When the recurrent state is carried into a later window, CHARTA leads by 2.5 points at eight pairs and 12.2 points at sixteen, the two loads beyond training. The lower rows, with every pair still in view, show the same pattern; their sixteen-pair values are the ones reported in Table 1.

Two writes per token. In the long-gap recall, memory-load, and person–code binding experiments, CHARTA applies two delta writes per token, following DeltaProduct (Siems et al., 2025). All other experiments use one. Table 6 separates this second write from CHARTA's other components at sixteen pairs. With every pair in view, CHARTA with a single write per token, at 2.10M parameters against DeltaNet's 2.07M, still exceeds DeltaNet (90.7 against 82.3 over three seeds), and with two writes per token CHARTA exceeds DeltaProduct at 2.39M parameters, both in view (96.7 against 90.0) and when the recurrent state is carried into a later window (94.2 against 85.8). Carried, single-write CHARTA reaches 79.0, level with DeltaNet's 82.0 within seed variation. With writes per token matched, CHARTA therefore leads in every comparison except carried recall with one write, where the second write is needed. At matched size and writes per token, CHARTA also halves held-out perplexity on these runs, with 41.4 against 80.2 for DeltaNet and 31.2 against 65.5 for DeltaProduct.

B Training Long-Gap Recall and Person–Code Recall Together

The preceding experiments train long-gap recall and person–code recall separately. We next test whether one model can learn both by training on a 50/50 mixture of raw stories and automatically generated training examples. The answer keys are produced by the script that generates each example. No human labels are introduced and no grader remains at test time.

At matched compute, the mixed model has higher perplexity because the generated examples occupy optimisation steps. Matching the number of *language-modelling tokens* reduces the difference to 0.03 perplexity. The jointly trained model retains 98/95/86% recall at gaps of 320/512/768 tokens and reaches 94–96% across tests containing 1–16 simultaneous person–code pairs, including queries beyond its window.

A pure-text control uses the same architecture and compute without the generated examples. It remains at chance on every evaluation task (3–7%). The generated examples account for the acquired skills; general language training alone does not pass these synthetic evaluations.

Pairs held	1	2	4	8	16
Carried across a window					
CHARTA	100	100	100	**99.8**	**94.2**
DeltaNet	100	100	99.8	97.3	82.0
All pairs in view					
CHARTA	100	100	100	**99.8**	**96.7**
DeltaNet	100	100	100	97.3	82.3

Table 5: Person–code recall (%) with 1–16 simultaneous pairs. All rows are three-seed means. In the upper rows the query arrives after the recurrent state is carried into a later 256-token window at a 512-token gap; in the lower rows every pair remains inside the input. Both models trained with at most four pairs, so the 8 and 16 columns are beyond the training load. Chance is 6.25.

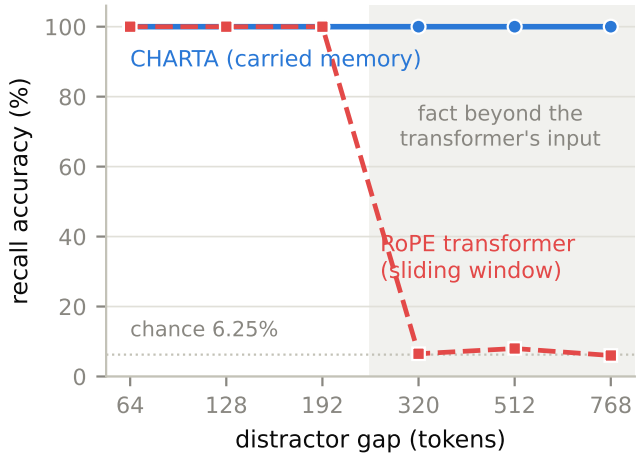

Figure 2: Once-stated fact recall as the distractor gap grows. Both models are perfect while the fact is inside the transformer's window; beyond it the transformer falls to its measured chance rate while the recurrent boards carry the fact across three window boundaries.

C Effects of the Added Training Examples

Part of the entity result comes from the architecture and part from what the model is taught. The following experiments measure these effects separately.

Learning a new pair from one statement. A set of training examples uses invented words that do not occur as whole words in the corpus. It takes recall of a never-seen pair after one statement from chance (25%) to 100% at this scale. The result is replicated across seeds and remains after training on the complete corpus mixture. Generalisation follows the training variation exactly: novel nouns in trained sentence shapes transfer (38% against a 10% control) while novel addressing styles do not. The transfer disappears when the curriculum is diluted fourfold in ordinary text, while performance on the trained forms remains intact.

Training on fact changes. Teaching new associations alone produces a model that answers questions after a fact changes with a container's *previous* occupant 100% of the time. Adding training stories that first state a fact, then move, swap, remove, or replace it, and finally ask again cuts this old-answer rate from 100% to 22% and lifts entity tracking to 57.77 from a previous best of 55.35. The gain appears in exactly the predicted subscore: the entity-tracking task's *move-contents* subscore rises from 41 to 67.

These experiments separate the effects of the two sets of training examples. Person–item examples teach new person–item pairs. Without examples of fact changes, the model always returns the previous occupant after a change. Adding change examples reduces this old-answer rate and im-

Model	Writes	Params	Perplexity	In view	Carried
CHARTA	2	2.41M	$\mathbf{31.2}\pm\mathbf{0.7}$	$\mathbf{96.7}\pm\mathbf{4.2}$	$\mathbf{94.2}\pm\mathbf{4.0}$
CHARTA, single write	1	2.10M	41.4 ± 2.7	90.7 ± 2.1	79.0 ± 7.9
DeltaProduct	2	2.39M	65.5 ± 1.5	90.0 ± 2.2	85.8 ± 3.6
DeltaNet	1	2.07M	80.2 ± 1.9	82.3 ± 3.6	82.0 ± 4.1

Table 6: Person–code recall (%) at sixteen pairs and held-out perplexity on the distractor text, as three-seed means. *Writes* is delta writes per token. *Carried* uses a 512-token gap across a window boundary. For one DeltaProduct seed, the streaming check used a relative tolerance of 0.002 instead of 0.001 (measured 0.0011, with identical next-token choices).

proves entity tracking. Object-file heads make the initial address stable, while the change curriculum teaches the model to replace an obsolete association.

D Revision After a Fact Changes

Section 6 shows that object-file heads make a moved fact slow to update. In the relocation test, an item moves to an empty box and the model is asked what that box now contains. We tested seven mechanisms for removing or suppressing the old fact when an item moves. None improved relocation accuracy, even though a linear decoder still recovers the updated fact from the boards. Proofreading, address orthogonalisation, an influence-conditioned write gate, and hard winner-take-all reads also give no tracking gain under the tested configurations.

The updated fact is therefore stored, but the model does not consistently select it when answering. The write succeeds, and the remaining error occurs when the model chooses which stored information to use for its answer. We call this a read-arbitration problem.

Transfer across entity and grammar tasks. We test whether training examples for one kind of operation help with another. Training examples in which items are moved, swapped, removed, and replaced raise the entity-tracking score by 32 points and improve each of those operations. Training examples of property inheritance, the operation that COMPS tests, do not improve COMPS. The gain therefore stays with the operations that were trained and does not extend to an unrelated task.

Using previous errors as feedback. We test whether the model can learn from its own earlier errors. In the first method, the model proposed several alternative ways of writing to its boards, and each proposal was scored by how well the model then predicted the training text. A separate choice head was trained to prefer the higher-scoring proposal within each group. Its choices were no better than chance, and the complete method performed worse than a control trained for the same wall-clock time. One simpler addition did help when tested alone. A learned vector summarising the prediction errors in the previous text window was added while processing the next window. It lowers perplexity by 2.9 against a control with the same samples, number of steps, and wall-clock time. This is a single seed and requires replication.

E Scope of Board Decoding and Transplantation

This appendix states what the decoding and transplantation experiments show about the boards, and what they do not show.

What causal transplantation shows. In transplantation, the model reads two stories, a donor and a recipient, and all boards from the donor are copied into the recipient before the question. This changes the recipient's answer to the donor's value. A separate attempt to construct an edit by hand, by applying the raw delta-rule update to token embeddings, changed answers no more often than the unrelated-box control. Transplantation therefore shows that boards the model itself produced carry the answer. Constructing new board contents by hand remains an unresolved problem.

What the linear decoder shows. Section 5.3 reads the boards with a fitted linear decoder, not by looking at them. No single board matrix contains a readable list of facts. The decoder is fitted on some item words and still recovers the locations of item words it never saw during fitting, which is why the downstream classifier in Appendix M is also given learned board reads. This shows that item locations are stored in a form a linear decoder can read. It does not show that a person can interpret every number in a board by looking at it.

How object-file heads affect revision. Stable addresses are more difficult to overwrite. In the relocation test, an item moves to an empty box and the model is asked what that box now contains. The full model gives the moved item on 0.5% of stories, against 81% after removing object-file heads and 42.0% for matched DeltaNet. In-place replacement, where a new item replaces the old one in the same box and so overwrites the address the item already holds, is answered at 99.7% by the same checkpoints (Appendix S.1). The present object-file design therefore gives a stable first association at the cost of slower relocation. Figure 3 traces this trade-off sentence by sentence.

F Tested Architectural Additions

Adding forgetting mechanisms. Most related recurrent memories use some form of decay. We tested four such mechanisms in CHARTA. An additional additive memory reduced long-range accuracy by 9.4 points and caused replaced facts to reappear. An exponential-moving-average memory produced no measurable change. The learned gates of a fast-decaying memory remained near their initial values. A cache that removed old tokens performed worse than a comparison cache that retained the tokens the model found least predictable. None of the four improved the tested result.

Other changes tested. Table 10 reports matched experiments on ten additions. Four introduce forgetting. The others test a second pass through the model, exact-token buffers, writes scaled by how unexpected each token is, and self-supervision.

Changes to addressing or auxiliary supervision of current story facts improve at least one target elsewhere in the paper. None of the additional memories in Table 10 improves its target under the tested configuration.

Training on reversed text. Training on the same tokens read right to left, inside the same 10-epoch budget (optimisation steps reduced from 18,790 to 15,032 so total token exposures are unchanged), does not materially change perplexity (33.42/33.53/33.72 against 33.53) but reduces entity tracking by 7.6 points across three seeds and improves no other column. We did not retain it. At the initial seed it looked like a gain, which is why results reported only at a matched seed are identified as such.

Multi-seed replication. The reported configuration is replicated across three seeds: zero-shot mean 53.95 ± 0.43, entity tracking 59.97 ± 0.84, with the box-tracking test at 100% in all three, and the board-assisted classifier suite averages 64.51 ± 0.58 over three fine-tuning seeds. Replication changes the interpretation of two results above. After the second event-router seed, the mean difference falls within the observed seed variation. The backward objective's entity reduction becomes clear across three seeds.

G Effect of Training Data on Perplexity

This gap follows from what each model keeps. Attention retains direct access to the preceding token sequence, whereas a fixed-state recurrent model compresses that sequence into its state. CHARTA structures the compressed state so that selected task-relevant information remains persistent and accessible; it does not aim to preserve every recent token. We test the source of the perplexity difference by asking each model to recover the exact token observed 8, 16, or 32 positions earlier. CHARTA reaches 14%, 8%, and 5%, and additional training does not raise these values. The boards do not preserve an exact copy of the recent sequence, while Section 5.1 shows that a queried fact remains available after its source words leave the input.

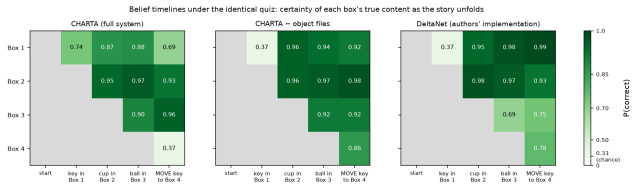

Figure 3: Binding and relocation with object-file heads on one box story. After every sentence, the model is asked what each box contains, and brightness is the probability it gives to the correct content among three candidates. The last sentence moves the key from Box 1 to Box 4. CHARTA (left) binds a fact on first mention and retains it, but is slow to adopt the move. Removing only object-file heads (centre) weakens first binding and adopts the move more readily. Matched DeltaNet is shown at right.

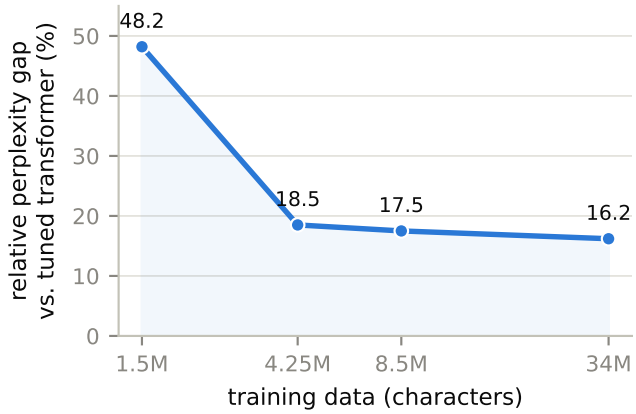

Figure 4: The in-window perplexity gap against a matched transformer narrows with data and flattens in the mid-teens.

H Hyperparameters

Six blocks, $d_{\text{model}} = 384$, six heads (two of them object files), head dimension 64, vocabulary 8,192, streaming window 256, no positional embeddings, short causal convolution of width 4, eight learned write channels. Each block is pre-norm with LayerNorm; the MLP has width $4d_{\text{model}}$ with GELU; the per-head normalisation after the board read is one LayerNorm of size 64 shared by all heads; the delta recurrence runs as a chunk-parallel scan with chunk 128, which is identical to the token-by-token update. Pretraining uses AdamW at a peak rate of 2×10^{-3} with a cosine schedule, batch 32, 18,790 steps, which is ten passes over the 10M-word corpus. The curriculum mixes a fixed synthetic set of entity-tracking training stories (11,345 stories, 1.0M words, inside the 10M budget) at a fraction of 0.3.

Fine-tuning uses the BabyLM pipeline at a learning rate of 10^{-3}, batch 32, ten epochs, sequence length 512, left padding, last-token pooling, with the checkpoint retained on each task's validation metric.

I Matched Architecture Controls

MAD tasks. Compression reconstructs a sequence from one summary token, in-context recall (ICR) recovers shuffled key–value pairs, fuzzy recall uses keys and values of one to three adjacent tokens, noisy recall adds distractors, memorisation learns a fixed key–value dictionary, and selective copying (SelCopy) reproduces marked tokens in order while ignoring distractors (Poli et al., 2024).

MAD task examples. Below, letters are key tokens, digits are value tokens, `x` and `y` are noise tokens, _ is a blank, and # is a special task token. Real sequences have no spaces or separators.

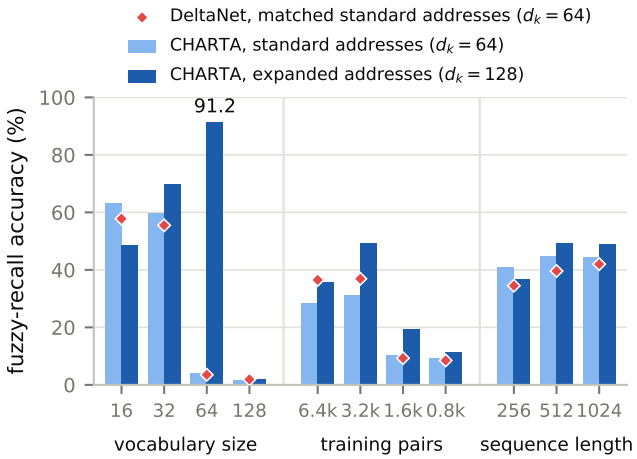

Figure 5: MAD fuzzy recall. Each point is the best of the MAD six-setting optimiser sweep. Light and dark bars use CHARTA with 64- and 128-dimensional addresses, and diamonds use DeltaNet run with the same setup and optimiser sweep, using 64-dimensional addresses.

```
In-context recall:  A 7 C 3 D 5 ...  A → 7
Noisy recall:       A 7 x C 3 y y D 5 ...  A → 7
Fuzzy recall:       A B 7 C 3 9 D E F 5 ...  D E F → 5
Memorisation:       A # C # D # → 7, 3, 5
Compression:        4 9 2 7 5 # → 4 9 2 7 5
Selective copying:  4 _ _ 9 _ 2 _ 7 # → 4 9 2 7
```

In fuzzy recall, keys and values are one to three tokens long, and test queries always use three-token keys. In memorisation, the dictionary is the same in every sequence and values never appear in the input, so it must be stored in the weights. In compression, the model must reconstruct the sequence from its representation at the final token.

DeltaNet MAD result. Every row of Table 4 except DeltaNet's follows the MAD protocol: the MAD authors' own for Transformer++, Mamba and GLA, and ours for CHARTA. The DeltaNet row is the figure its authors report, produced under their own training setup. Their appendix specifies batch size 32, four times the optimisation steps used by the MAD protocol, and a learning-rate sweep reaching 3×10^{-3}. This is higher than any learning rate used for the other rows. Table 4 uses their published result and identifies the protocol difference.

Address width and vocabulary. At vocabulary size 64, CHARTA with 128-dimensional addresses scores 91.2% on fuzzy recall, against 3.6% for DeltaNet with its standard 64-dimensional addresses (Figure 5). On MQAR, a standard multi-query recall benchmark (Arora et al., 2024), CHARTA also follows DeltaNet's published curve as model width grows, reaching 98.1, 99.3, and 99.9% at widths 64, 128, and 256.

Trained on our corpus with our training command, the size-matched controls reach held-out perplexities of 32.89 for KDA and 36.45 for Gated DeltaNet, against 33.53 for CHARTA. Each control replaces every block's mixer with that architecture's own layer from the `flash-linear-attention` library (Yang & Zhang, 2024), leaving embeddings, norms, the language-model head, the trainer and the curriculum identical. DeltaNet sets its head dimension to d_{model}/H. Gated DeltaNet and KDA instead use fixed default head dimensions (256 and 128), which at our width would give mixers of 38.8M and 19.3M parameters against CHARTA's 13.9M. We therefore set their head dimension and value expansion so that their mixers have 14.8M and 14.5M parameters. In every architecture, padded positions are prevented from writing to the recurrent memory, the recurrent counterpart of an attention mask. Without this, the comparison would measure the effect of that fix rather than the architectures.

J BLiMP Error Analysis

CHARTA scores below the archived entry on BLiMP. We separated categories that improve with additional training from those that do not. The base model scores 21.8 on the latter group, and additional training does not change their mean. We tested corrections based on checkpoint outputs and grammar rules.

Subtracting early-checkpoint scores. Subtracting a scaled output from the model's one-epoch checkpoint moves these categories from 21.8 to 57.1 with *zero* additional training. Oracle category-level selection between the two checkpoints would reach 72.25; a split-half cross-validated fixed combination reaches 65.9. The one-epoch checkpoint assigns different scores to some errors that remain in the final checkpoint.

Filtering BLiMP answers with grammar rules. The filter contains ten stateful rule families over roughly 200 closed-class words and plural morphology. They cover agreement, gap bookkeeping, NPI licensing, conditional environments, and island constraints. An audit on natural speech found a 7% flag rate and found the flags largely correct on disfluent input. Applying the filter leaves entity tracking unchanged to the second decimal and raises BLiMP from 63.8 to 69.4. Combining it with the checkpoint method reaches 71.9, so the two methods correct partly different errors.

As the base checkpoint improves, the grammar filter adds less while the combined score remains near 71. Fewer of the remaining errors match the implemented rules.

Rule-derived supervision. Training the network with rule-derived token-level unlikelihood, with zero new text, raises NPI licensing to 99.5%, replicated across seeds. The same approach does not improve agreement across a distractor, even with full-span signals. The tested local rule is learned; the tested relational rule is not.

K Full Component Ablation

Table 7 reports all eight combinations of the organiser, object-file heads, and event router under the final training curriculum. Each configuration keeps or removes each of the three components, and every other setting is unchanged. *Mean* is the zero-shot mean over six tasks, the four shown and EWoK and COMPS. Rows are a single seed unless marked.

L Additional Analyses

The analyses below complement Sections 5.3 and 6; the setting and checkpoints are those of the corresponding section.

Which heads does the linear decoder use? Restricting the decoder to object-file heads gives 24.87%, close to chance, while the remaining heads give 50.78%. The object-file heads establish stable addresses and the contextual heads contain most of the information that the classifier recovers for unseen item words. This measurement is specific to the fitted linear decoder; it does not establish that the object-file boards contain no information.

Similar scores, different internal state. Final tracking accuracy does not expose these internal differences. On quizzes with static item locations, facts surrounded by repeated mentions, and box contents, CHARTA scores 100/100/100 and matched DeltaNet scores 99.5/100/99.5. Yet the linear decoder recovers 49.0% from CHARTA's boards and 26.1% from DeltaNet's.

Scarce and abundant training data. Training data can also reduce the architectural difference. With abundant BabyLM entity data, DeltaNet reaches 60.21 and CHARTA reaches 59.97 ± 0.84. When entity data is scarce, CHARTA reaches 44.65, compared with 34.41 for DeltaNet and 16.19 for matched attention. CHARTA leads when training examples are scarce; with abundant data, CHARTA and DeltaNet reach similar entity scores.

Configuration	Entity	BLiMP	Supplement	GlobalPIQA	Mean	Moved fact
organiser + object files + router	60.76	66.31	58.26	39.64	54.44	0.5
no router (2 seeds)	62.13	66.22	57.55	32.71	53.68	19.3
no object-file heads	60.30	67.33	58.27	38.11	54.33	**81.0**
no organiser	52.89	66.94	56.28	36.62	52.44	18.0
organiser only	52.09	67.00	57.53	37.14	52.73	23.0
object-file heads only	56.42	65.07	58.78	34.68	52.91	36.0
router only	63.05	66.95	57.56	36.12	54.28	61.5
none of these three	58.43	67.17	57.20	34.67	53.35	64.5

Table 7: All eight combinations of the organiser, object-file heads, and event router under the same final training curriculum. Each row is a complete evaluation. *Moved fact* is the relocation test, in which an item moves to an empty box and the model is asked what that box now contains. Removing the organiser costs 7.9 points of entity tracking. Removing object-file heads leaves entity tracking intact and raises moved-fact accuracy from 0.5% to 81.0%. The change from removing the router is within the variation observed across its two seeds.

M Downstream Classification with Board Reads

If the boards hold task-relevant information that the final hidden vector does not pass on, a classifier that also reads the boards should perform better. We test this on the seven (Super)GLUE tasks of BabyLM, fine-tuning with the standard pipeline (Appendix H). The *last-token classifier* receives only the final hidden vector, as in the standard evaluation. The *board-assisted classifier* also receives eight learned reads $M^\top a_r$ from the final boards. A zero-initialised projection makes the two classifiers identical before fine-tuning, and all other conditions are unchanged.

Across three paired CHARTA seeds, the board reads raise the seven-task mean by 1.47 ± 0.31 points, with a gain in every run (Table 8). The largest matched-seed gains are on QQP (62.69 to 67.15), MultiRC (57.55 to 61.67), and RTE (56.12 to 59.71), while the same change gives DeltaNet $+0.23$. MultiRC separates the classifier inputs most clearly, because predicting only the negative class already gives 57.55 accuracy with zero F1. CHARTA's board-assisted classifier exceeds this in all three seeds, averaging 62.13 accuracy and $\mathrm{F1} = 0.489$, while every other tested classifier input has F1 at or below 0.217. For KDA and Gated DeltaNet, whose boards decode near chance (Table 2), we ran the board-assisted classifier on MultiRC, the task that separates the inputs most clearly, which is why we do not report a mean over the seven tasks for them. KDA stays at the majority-class baseline (57.67, F1 0.008), and Gated DeltaNet returns to it in two of three seeds (57.55–58.21, F1 0.000–0.217).

N Evaluation Task Examples

This appendix reproduces the exact formats of the synthetic evaluations, taken from the generating code, so one can judge what the model did and did not see.

N.1 Object tracking

Examples are built from four story-update templates over eight people, eight places, and a tracked object:

```
Alice has the red_key.
Bob takes the red_key from Alice.
Clara takes the red_key from the box.
Derek gives the red_key to Mina.
Omar puts the red_key in the drawer.
```

Training examples contain 1–8 such events; evaluation examples contain 9–12, so every evaluated length is unseen. The query is an eight-way choice:

	CHARTA		DeltaNet	
Task	last	+boards	last	+boards
BoolQ (acc)	67.03	**67.58**	64.71	65.69
MNLI (acc)	48.94	48.06	**50.63**	47.90
MRPC (F1)	81.76	**82.69**	82.49	82.04
MultiRC (acc)	57.55	**61.67**	57.59	57.55
QQP (F1)	62.69	**67.15**	64.30	66.22
RTE (acc)	56.12	**59.71**	56.12	56.12
WSC (acc)	63.46	63.46	61.54	**63.46**
Mean	62.51	**64.33**	62.48	62.71

Table 8: (Super)GLUE classification. "Last" uses the final hidden vector; "+boards" adds eight learned board reads. CHARTA gains 1.47 ± 0.31 across three paired seeds; DeltaNet uses the matched seed.

```
Options: A=Alice B=Bob C=Clara D=Derek E=Mina F=Omar
G=Priya H=Ravi
Question: Who has the red_key? Answer:
```

The model answers through its language head; chance is 12.5%. A location variant asks `Where is the red_key?` over eight places. During training, the grader is asked for every current holder and location after each completed story update.

N.2 Transitive deduction

An ordering over entities is sampled; only *adjacent* comparisons are stated, in shuffled order:

```
C is taller than D. A is taller than B. B is taller than C.
Question: Is A taller than D? Answer:
```

Queried pairs span 1–7 hops; answering a distant pair requires composing the stated facts. Chance is 50%. The grader is asked for each person's position in the complete eight-person order. No queried pair at evaluation distances appears in training.

N.3 Secret-word recall and concurrent binding

The fact and query templates, sixteen names, and sixteen candidate codes are, verbatim from the generator:

```
fact:  "{name} secret word {code} .  "
query:  "{name} secret word"
names:  Alice Bob Clara Derek Mina Omar Priya Ravi
        Nadia Tomas Uma Victor Wendy Xena Yusuf Zoe
codes:  zorp blick quaz fendle grumt plisha vekt norlu
        brimzo skarn tulve wexid yamber crolt dulps mavrik
```

The codes are nonsense strings that never occur in natural text, so the distractor stream (real TinyStories text) carries no hint. In the long-gap recall experiment, an example states one pair, inserts D story tokens, and then queries that person. Both models train only on $D \leq 192$; evaluation extends D to 768. In concurrent binding, an example states several distinct person–code pairs and queries one of them. Training uses at most four simultaneous pairs; the crowding evaluation uses up to sixteen. Table 1 reports the in-window condition, in which all facts remain visible. In both experiments, the model's likelihood is compared across all sixteen codes, so chance is 6.25%.

N.4 Box stories

Three evaluations use stories about items in numbered boxes. Answers are scored by comparing the model's log-probabilities for a fixed set of candidate continuations.

Location decoding (Section 5.3). Four items are placed in five boxes, and up to three are moved into an empty box:

```
Box 1 contains the cake, Box 2 contains nothing, Box 3
contains the bell, Box 4 contains the rope, Box 5 contains
the stick.  Move the bell from Box 3 to Box 2.
```

The linear decoder must place the bell in Box 2 using only the final boards. Item words are disjoint between fitting and testing.

Relocation (Section 6). Three or four boxes are filled, and then one item moves to a new box:

```
Box 1 contains the cake, Box 2 contains the bell, Box 3
contains the rope.  Then the bell was moved to Box 4.  Box 4
contains
```

The model must choose *the bell* over the other items in the story. The same story ending "Box 2 contains" measures whether the old location still wins. Figure 3 traces one such story, asking after every sentence what each box contains.

Transplantation (Section 5.3). Four boxes are filled, and the same story is run twice with one box changed:

```
Recipient: Box 1 contains the cake, Box 2 contains the bell,
Box 3 contains the rope, Box 4 contains the stick.
Donor:     Box 1 contains the cake, Box 2 contains the coin,
Box 3 contains the rope, Box 4 contains the stick.
```

After the recipient's boards are replaced with the donor's, the model completes "Box 2 contains the" and should prefer *coin* over *bell*, while Box 3 should still give *rope*. In the wrong-value control, the donor's Box 2 holds a third item such as *ring*, which the model should then prefer. In the different-box control, only the donor's Box 3 changes, and the Box 2 answer should stay *bell*.

O Additional Negative Results

For each mechanism, we fixed the proposed operation, predicted benefit and failure test before running it. This appendix records the design and the observed outcome.

Running the block stack twice. We passed the output of the complete block stack through the same stack again before answering. Tracking fell from 100% to 32%. Layers trained for one pass changed hidden vectors and boards that already produced the correct answer.

Refining a copy of the board before answering. We trained a module to modify a copy of M before the model answered while leaving the original boards unchanged. It produced no improvement on tracking or deduction. Both tasks already produce their answers during the first reading pass.

Grading current facts at every token. We applied the auxiliary targets for all current story facts at every token instead of only after a completed story update. Performance was consistently below update-only grading because some positions occur before the sentence has established the next fact.

Adding a second memory by addition. We formed a second memory by adding successive states from M. Previously replaced facts reappeared, and long-range accuracy fell by 9.4 points.

Adding an exponential-moving-average memory. We formed a second memory as an exponential moving average of M. It produced no measurable change and did not recover information lost from M.

Scaling writes by how unexpected a token is. A token has high *surprisal* when the model assigned it low probability before seeing it. We tested whether these less predictable tokens should write more strongly to the board. We first applied the proposed scaling rule during evaluation without retraining the model. The trained write gate produced similar values across most tokens. Increasing the relative strength of high-surprisal writes did not improve performance, while reducing writes from low-surprisal tokens made it worse. We did not run a full training experiment. In this setting, predictable tokens still contributed information used by the recurrent boards.

Backpropagating across window boundaries. Extending gradients across window boundaries during streaming training produced no measurable change at any tested range.

Adding a fast-decaying memory. We added a fast-decaying memory beside the delta memory to retain recent tokens. Its learned per-head gates remained near their zero initialisation.

Adding a 60-token exact buffer. We added an exact token buffer with a learned access policy. The learned weights gave less access to object-file heads and more to contextual heads, but the buffer did not improve the target metric.

Matching repeated mentions with a contrastive loss. We trained a contrastive objective on mentions of the same entity at different distances. It was intended to replace auxiliary targets derived from the known story answers, but it did not produce the predicted improvement at longer distances. A small gain at wide contexts came from another part of the configuration, so no improvement is attributed to this objective.

P Training Details

The model is a six-block pre-norm decoder with width 160. Each block has five heads of dimension 32 and one 32×32 delta memory per head. Two heads compute write addresses and read cues from raw token embeddings; these are the object-file heads. A width-4 causal depthwise convolution is applied to contextual addresses, contents, and read cues. We use dropout 0.1 and AdamW with weight decay 0.1 under a warmup-plus-cosine schedule. Language-modeling runs use 256-token windows; the final tracking configuration uses learning rate 2×10^{-4}; the grader loss weight is 0.5 throughout. The corpus is TinyStories. The total parameter count is approximately 2M at this configuration; the 7.8M configuration widens the model. In the MAD fuzzy-recall comparison, expanding 64-dimensional addresses to 128 dimensions while retaining 64-dimensional content adds approximately 65,000 parameters. Full per-experiment configurations, protocols, and outcomes are recorded in the repository's experiment log, from which every number in this paper is drawn.

Q Matched DeltaNet Configurations

Because the published DeltaNet row's configuration is unspecified, we ran DeltaNet's own implementation through our setup with the same normalization, standard address width, optimiser sweep, checkpoint selection and scoring used by standard-address CHARTA. The two models score 29.6 and 30.6, respectively. We then doubled DeltaNet's address width through its own `expand_k` interface: its mean falls to 26.8, while expanded-address CHARTA reaches 42.0 under the matched LayerNorm comparison and 43.7 under the main table's RMSNorm setting. At vocabulary size 64, expanded-address DeltaNet remains at 3.6%, whereas CHARTA reaches 91.2% (93.5% under RMSNorm). Thus address expansion alone does not explain CHARTA's gain. Per-configuration standard-address DeltaNet and expanded-address CHARTA results appear in Table 9.

For reference, CHARTA run entirely at the setup defaults (RMSNorm blocks, four heads of width 32, no key expansion) scores 44.5 / 23.9 / 99.8 / 51.1 / 99.9 / 99.8 across the six tasks (mean 69.8). The gap to the main-table row (+19.8 on fuzzy recall, +1.8 on memorization) is the contribution of the configuration findings in the main text.

A LayerNorm variant of the old head geometry scores 52.9 on memorization, above the setup-default 51.1 that the main table reports; the main table keeps the RMSNorm value so that every entry in its row shares one normalization.

Fuzzy config	DeltaNet ctrl	CHARTA	Δ
vocab 16	**0.578**	0.486	−9.3
vocab 32	0.555	**0.696**	+14.1
vocab 64	0.035	**0.912**	+87.8
vocab 128	**0.019**	0.018	−0.1
train pairs 6400	**0.365**	0.357	−0.8
train pairs 3200	0.369	**0.492**	+12.3
train pairs 1600	0.093	**0.193**	+10.0
train pairs 800	0.085	**0.113**	+2.9
seq. len. 256	0.345	**0.367**	+2.2
seq. len. 512	0.396	**0.491**	+9.5
seq. len. 1024	0.420	**0.490**	+7.0
Column mean	29.6	**42.0**	+12.4

Table 9: Per-configuration fuzzy-recall bests (best of 6 optimiser settings each; bold marks the better model per row). The DeltaNet column uses standard 64-dimensional addresses; the CHARTA column expands the address dimension to 128 while retaining 64-dimensional content. Both use LayerNorm here. CHARTA's column mean is 42.0, against 43.7 for the main table's RMSNorm setting.

R Summary of the Negative Results

Mechanism	Design	Measured outcome
run stack three times	two further passes over the stack output	tracking 100%→32%
refine board copy	modify a copy of M before answering	no improvement
grade every token	predict all current facts at every position	below update-only grading
additive second memory	accumulate states from M	old facts reappear; -9.4 pts
EMA second memory	low-pass filter of M	no measurable change
scale unexpected-token writes	scale β by token surprisal	no gain; suppression hurts
backpropagate across windows	extend gradient across boundaries	no measurable change
fast-decaying memory	retain recent tokens in a side memory	gates remain near zero
60-token exact buffer	learned access to recent tokens	no improvement
contrastive mention matching	match repeated entity mentions	no distance improvement

Table 10: Ten matched architecture experiments. None improves its target at this scale and configuration. The added memories, longer gradient path, second model pass, surprisal scaling, and contrastive objective all produce either no change or a reduction in the measured result.

S Linear Board Decoder: Training and Evaluation

Each story places four items in five boxes, and the decoder must name the box that holds a given item, so chance is 20%. For a candidate box b and an item word w, the decoder receives the feature $[(M^\top q_b) \odot v_w,\ (M^\top k_b) \odot v_w]$. Here M is a block's board, q_b and k_b are that block's read cue and address for the box word, and v_w is its bounded content vector for the item word. The decoder is fitted separately for each block, and the block with the best validation accuracy is used. No item word used in fitting appears in testing, so memorising particular item words cannot produce the result. Results are averaged over three classifier seeds, and the same code and split are used for every architecture.

S.1 Previous Content After a Replacement

We test two distinct questions after correcting a value. The first is behavioural: can CHARTA's own language-model head still state the current content? The second asks what remains recoverable from the boards: can a newly fitted linear decoder identify either the current or the previous item?

For example, one held-out story says:

> Box 1 contains the cake, Box 2 contains the stick, Box 3 contains the brush, Box 4 contains the bell. The bell in Box 4 is replaced with a rope. Nothing else moved.

The behavioural test appends "Box 4 contains" and asks the frozen model to score four complete continuations: *the bell*, *the cake*, *the rope*, and *the stick*. The correct answer is *rope*. This is an ordinary model prediction: no board decoder is fitted, and each continuation is scored by its mean next-token log probability.

The decoding test uses the same story but does not ask the language-model head a question. Instead, for each candidate word it computes the board feature $[(M^\top q_b) \odot v_w, (M^\top k_b) \odot v_w]$ defined in Appendix S. A newly fitted linear decoder assigns one score to each candidate. One decoder is trained to select the current item (*rope* in the example); a separate decoder is trained to select the previous item (*bell*). These decoders test whether each relation is exposed in a simple, transferable linear form. They do not replace CHARTA's own word-prediction path.

Four-box stories replace one item once and present four candidate items. Decoder fitting, the choice of block and regularisation, and final testing use 800, 240, and 400 stories, with no item words in common. The language-model result is measured on those same 400 final-test stories. CHARTA remains frozen throughout.

We check the previous-item decoder with two controls. Setting the board feature to zero gives 24.3% at every checkpoint, which is chance. Giving each story another story's boards, over 150 random pairings per checkpoint, gives mean and maximum accuracies of 25.6 and 31.8% for checkpoint 1, 25.2 and 31.0% for checkpoint 2, and 25.3 and 30.5% for checkpoint 3.

Checkpoint	LM current	Linear current	Linear previous
1	99.0	99.5	93.5
2	100.0	90.5	57.3
3	100.0	55.7	79.2

Table 11: Two distinct post-replacement tests (%); chance is 25.0. "LM current" is CHARTA's ordinary answer accuracy on 400 stories. "Linear current" and "Linear previous" are separate linear board decoders, each averaged over three classifier seeds. Rows are three independently trained checkpoints.

The ordinary language-model head answers the current-content question at $99.7 \pm 0.6\%$ across checkpoints (mean $\pm$ sample SD). Thus the 55.7% entry in Table 11 is not a failure to track the current item: it is the transfer accuracy of a deliberately restricted new linear decoder on checkpoint 3. CHARTA's trained output path may combine the recurrent boards with the final hidden representation through nonlinear layers, whereas the separate classifier receives only one specified linear-in-board feature.

The previous-item decoder (93.5, 57.3, and 79.2% in Table 11) exceeds chance and the maximum of every shuffled-board control at all three checkpoints. Together, the two tests show that CHARTA reliably uses the corrected current value while the complete collection of boards after replacement still exposes information about the previous value to a separate simple classifier. This does not establish that one shared classifier recovers both, that the ordinary language-model head answers a historical question without explicit teaching, or that the previous item remains at the address used for the current value. The historical trace may be distributed across other heads, blocks, or directions selected by the event router.

T LIMITATIONS

The language-model evidence comes from controlled experimental scales and one 10M-word track whose curriculum includes synthetic entity practice. Whether the same linear board decoding persists after substantially larger-scale training is left to future work. The leaderboard comparison uses the archived snapshot dated in Table 3 and its filtered entity metric.

Replication is matched to the cost of each experiment, as Section 4 records. The BabyLM configuration, the board decoder, the board-assisted classifier, the deduction comparison, and person–code recall for CHARTA and DeltaNet use three seeds, and Appendix F reports the multi-seed replication. Table 6 gives the seeds behind each person–code comparison. The Transformer++ teaching control, the DeltaNet board-assisted classifier, and the individual component-ablation rows use one matched seed. Where a comparison comes from one seed, conclusions are drawn only from gaps that are large relative to the variation observed in the replicated runs, such as the teaching control and the relocation test, and small differences between ablation rows are interpreted as being within variation. Seed variation is especially pronounced for RTE and MultiRC (Bui et al., 2025; Reimers & Gurevych, 2018), and both CHARTA classifiers are replicated for these tasks. The entity-tracking advantage holds when data is scarce (Appendix L). Object-file heads slow the relocation of a moved fact; Section 6 reports the trade-off.