WHAT DOES A MODEL LEARN FROM A CHANGING WORLD?

Anonymous authors
Paper under double-blind review

ABSTRACT

Models absorb world knowledge during pretraining, but they do not see a static snapshot of the world: pretraining data spans many time periods, showing facts change over time. In a controlled setting, we first ask whether models simply memorize each time period separately, or learn how knowledge evolves across them. We find the latter: compared to a model trained on a single time period, models trained on multiple periods (a) better preserve historically constant knowledge and (b) more efficiently learn updates that follow historical patterns. We then find that the order in which facts are presented determines how they are stored. When facts are presented chronologically, they are stored in individual slots, so an update changes only the facts it trains on. When facts from different time periods are instead shuffled together, models use class-based storage: facts that have changed together, such as athletes who switch teams together, share a representation, so an update to one of them spreads to the others. We find evidence of class-based storage in OLMo-2-7B, whose pretraining data, like our shuffled setting, mixes documents from different time periods. Beyond the known tendency of updates to spread among facts that share a relation and answer, we find correlational evidence that updates spread further in groups of facts whose answers changed more during pretraining. Our results show that models adapt to the changes they observe during training, and that the order in which they observe them shapes how, suggesting pretraining data ordering as a way to influence how models absorb new information.

1 INTRODUCTION

Models memorize world knowledge during pretraining (Petroni et al., 2019; Roberts et al., 2020), but they do not see a single snapshot of the world. Pretraining corpora span documents from different points in time, allowing models to observe how knowledge changes. This data shows some facts remaining constant, such as a person's birthplace, while others, such as their position or employment, change over time. This gives models the opportunity to learn about the evolution of knowledge, beyond simply memorizing individual facts.

One way to probe models' understanding of knowledge evolution is to examine how they respond when facts change. Prior work on fact editing shows that updating a fact can modify seemingly unrelated knowledge while failing to update its logical implications (Akyürek et al., 2024; Cohen et al., 2023). For example, changing the CEO of one company can inadvertently change the CEO of an unrelated company, while failing to update logical implications of the new CEO. These behaviors suggest that models do not always respond to updates in ways that reflect how knowledge changes in the world. This raises a fundamental question: can models take advantage of this temporal information to learn how knowledge evolves?

We investigate this question in a synthetic fact memorization setting. Some facts remain constant over time, while others change according to a predictable rule. We find that exposure to changing facts alters how models respond to later fact updates. Models learn to keep constant facts unchanged and to quickly absorb updates to changing facts that follow historically predictable patterns. These behaviors emerge only in models exposed to multiple time periods, indicating that they are learned from observing change in their training data. This need not happen: a model could memorize each time period separately without changing how it responds to later updates.

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.

Does the way models observe changing knowledge affect what they learn? Knowledge evolution has an inherent chronological ordering, yet pretraining typically shuffles documents from different points in time together. We study whether this difference in ordering matters in our synthetic setting. We find that presenting knowledge chronologically (ORDERED) causes models to store facts in **individual slots**: each fact is represented relatively independently, and the model learns to move it quickly to its expected next-time-step answer. Shuffling data from different time periods (SHUFFLE) instead produces **co-movement classes**, grouping facts that have historically evolved together (such as athletes who move between teams together). This difference is consequential: individual slots enable more isolated updates, while co-movement classes couple facts under subsequent updates.

We next investigate why the presentation of data changes how facts are stored. Early in training, ORDERED sees a consistent set of facts and memorizes them quickly using the relatively independent representations provided by random initialization. This creates largely individual slots, which the model learns to rewrite as it encounters subsequent time-steps. Under SHUFFLE, the model instead sees conflicting versions of each shifting fact from the beginning of training, and this interference delays memorization until shared representations have formed. As a result, shifting facts are memorized along shared directions, which carry updates from one fact to the others in its class.

We next turn from our controlled setting to a pretrained language model. We study OLMo-2-7B (OLMo et al., 2025), whose pretraining data, like our shuffled setting, mixes documents from different time periods, and whose intermediate checkpoints are public. Across nearly a thousand edits, we find correlational evidence that updates spread further in groups of facts whose answers changed more across pretraining checkpoints, controlling for the known tendency of updates to spread among facts that share a relation and an answer (Meng et al., 2022). Because we cannot vary how a pretrained model observed its data, whether these groups arise from temporal change remains an open question.

Neither form of storage is uniformly better, mirroring a trade-off between out-of-context reasoning and hallucination identified in prior work (Huang et al., 2026). Class-based storage lets an update to one fact generalize to the facts that have historically changed with it, which helps when that pattern continues but produces errors when it breaks (for example, when only one of several athletes who used to switch teams together is traded). Individual slots avoid such errors, at the cost of requiring each fact to be updated separately.

Together, our results show that models learn from observing facts change, beyond memorizing the facts themselves: exposure to multiple time periods changes how they respond to later updates, and the order in which they see those changes decides whether facts are stored individually or in classes. Our controlled experiments establish this causally, and our OLMo experiments show that the class-based coupling they predict also appears in a pretrained model. This points to the presentation of pretraining data as a lever on how models absorb new knowledge in continual learning scenarios.

2 RELATED WORKS

Continual Learning and Model Updating Keeping models up to date after training is a fundamental problem in deep learning. A central hurdle is catastrophic forgetting (McCloskey & Cohen, 1989; Parisi et al., 2019), wherein training on new tasks or distributions degrades previously learned capabilities. A large body of works have studied techniques for mitigating catastrophic forgetting including replaying data from existing capabilities (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017), regularizing movement from existing weights (Kirkpatrick et al., 2017), targeted parameter updates (Meng et al., 2022; Biderman et al., 2024), and leveraging on-policy data (Shenfeld et al., 2026). Another class of challenges involves ensuring that model updates reliably generalize to their intended scope. Cohen et al. (2023) demonstrate that existing fact update techniques often fail to produce consistent changes in model behavior. Existing works have proposed data-level interventions to ensure that model updates propagate to their logical consequences (Akyürek et al., 2024). Largely, existing research focuses on update-time interventions for enabling continual learning. In this work, we instead study how models *learn* from their training data how to respond to future updates.

Fact Storage in Transformers Several prior works have investigated the mechanism of fact storage in transformers. Geva et al. (2021) provide a construction viewing multi-layer perceptron modules as key-value memory. Meng et al. (2022) operationalize this structure to implement model editing

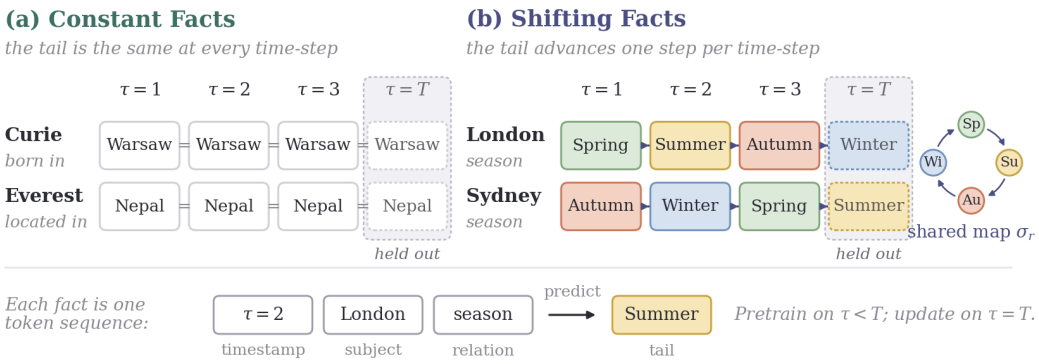


Figure 1: **Temporally Shifting Knowledge Graph Setting.** We study a knowledge graph-based fact memorization setting, where facts are represented as (timestamp, subject, relation, tail) tuples. Our setting consists of two categories of facts: **(a) Constant Facts** are invariant to the time dimension and observed with a constant answer. **(b) Shifting Facts** depend on the time axis: the answers evolve according to a predictable, cyclic structure across time. The final time-step $\tau = T$ is held out for the update.

via rank-1 model updates. Leveraging this structure, multiple works have studied the efficiency of fact storage and proposed additional constructions for improving the efficiency of the storage (Garcia et al., 2026; Nichani et al., 2024; Dugan et al., 2025). Recent works have challenged the purely associative view of transformer fact learning, observing that models can learn geometric structures that support implicit multi-hop reasoning and search (Noroozizadeh et al., 2026; Nishi et al., 2025). In this work, we study fact storage through the lens of continual training, uncovering two fact storage strategies that emerge in the presence of temporally changing data.

Model Training Dynamics An emerging line of work studies model training dynamics, particularly in the context of foundation models. Liu et al. (2026) demonstrate that language model pretraining tends to follow a predictable curriculum, composing previously learned, simpler capabilities throughout training. Wen et al. (2026) study the in-context generalization dynamics of language model pretraining, showing that capabilities can abruptly emerge and revert throughout training. Feng et al. (2026) study the dynamics across multiple stages of training, demonstrating that mixing post-training data into pretraining makes domain knowledge resistant to subsequent forgetting. In this work, we study how upstream training dynamics shape how models respond to downstream gradient-based updates.

3 A Controlled Setting for Studying Temporal Learning

3.1 A Time-Shifting Knowledge Graph

Constant and shifting facts. We study fact memorization in a synthetic knowledge graph over entities $\mathcal{E}$ and relations $\mathcal{R}$, where each relation r has an answer set (codomain) $\mathcal{A}(r) \subseteq \mathcal{E}$ (Ghosal et al., 2024). We extend the setting to be *time-varying* by partitioning the relations into a **constant** set C and a **shifting** set S. A fact is a (subject, relation, tail) triple, and we call the tail the fact's answer. For example, (Curie, born in, Warsaw) is a constant fact, while (London, season, Spring) is a shifting fact whose answer becomes Summer at the next time-step (Figure 1). In our experiments, subjects, relations, and answers are abstract entity tokens. The graph at time-step τ is

$$K(\tau) = \Big(\bigcup\nolimits_{r\in C}(\tau, h, r, a)\Big) \cup \Big(\bigcup\nolimits_{r\in S}(\tau, h, r, a_r(\tau))\Big),$$

where constant facts keep a fixed answer across time while shifting facts take a time-varying answer $a_r(\tau)$. All subjects, relations, answers, and time-steps are represented as separate tokens.

Predictable dynamics. We consider a setting where the evolution of shifting facts is predictable. For each shifting relation r we arrange its answer set on an ordered ring and let σ_r map each answer to the next position on the ring. A shifting fact (h, r) starts at a random position a at $\tau = 1$ and advances one position per step, so its answer at time τ is $\sigma_r^{\tau-1}(a)$; facts sharing a relation follow

the same map σ_r and differ only in their starting position (Figure 1). We pretrain on the first $T-1$ time-steps and hold out the final step $\tau = T$ for updating. Since T never exceeds the ring length, the held-out step never repeats a step seen during pretraining. Our main experiments (Figure 2(b)) graph with 300 entities, 8 relations, rings of 32 answers, and $T = 31$ time-steps. To investigate the training dynamics in Section 4.3, we consider a smaller scale setting with 100 entities, 8 relations, rings of 10 answers, and $T = 6$.

3.2 Updates and Evaluation

We model a knowledge update as continued training on the shifting facts at a held-out step $\tau = T$, where the shifting facts have advanced to their next positions. We characterize how each model takes on this update with three probes.

Stability. To measure how much prior knowledge a model preserves through an update, we record the fraction of *constant* facts it still recalls after continuing training on the held-out time-step. Concretely, we early stop the update training at the first point when shifting facts reach 95% accuracy (relative to their assignments at time-step $\tau = T$).

Update plasticity. We study how plastic models are to knowledge updates by measuring how many gradient steps are required to learn a future time-step's shifting facts to 95% accuracy.

Update coupling. We investigate whether knowledge updates spread across facts by updating a random fraction p of the shifting facts to their time-step T answer and measuring the percentage of held-out shifting facts that also move to their time-step T answer. Like the other probes, we early stop when the training accuracy reaches 95% and measure the time-step T accuracy at this point.

3.3 Training Conditions

We compare three pretraining strategies on the *same* facts, holding both the model and the update procedure fixed. In all three, the model is a 6-layer, 6-head decoder-only transformer with embedding dimension 192 (roughly 2.7M parameters), trained from scratch. The schemes are:

- **Ordered**: the model sees the time-steps in order, one block of T_{step} steps per τ; within a block the facts for that τ are shuffled.
- **Shuffle**: the model samples facts uniformly across all time-steps.
- **Snapshot**: the model trains only on a single time-step ($\tau = T - 1$), seeing a single snapshot of the world in time.

Ordered and Shuffle see identical facts across all time-steps and differ only in *order*. Snapshot sees only a single time-step and cannot observe any changes in knowledge, serving as a reference for a model that cannot learn temporal structure by definition.

4 Learning How Facts Change

4.1 Temporal Exposure Shapes Responses to Future Updates

We first ask whether observing the world change shapes how a model responds to future updates. Using the probes of Section 3, we compare the two temporally-exposed models, Ordered and Shuffle, against Snapshot, which sees only a single time-step and serves as a baseline for the effect of temporal exposure itself.

Stability. Through the held-out update, Snapshot recalls only about 0.68 of its constant facts, while both temporally-exposed models retain about 0.96, with Ordered and Shuffle indistinguishable (Figure 2(a)). Stability therefore comes from temporal exposure and arises regardless of the order of presentation.

Plasticity to future updates. We measure how quickly the different models learn shifting facts across future time-steps. Snapshot is slow at every offset, taking about four times as many steps as Ordered needs for the anticipated $\tau = T$ update (Figure 2(b)). Ordered absorbs the chronologically expected update in about 30 steps but slows when trained on updates for a time-step

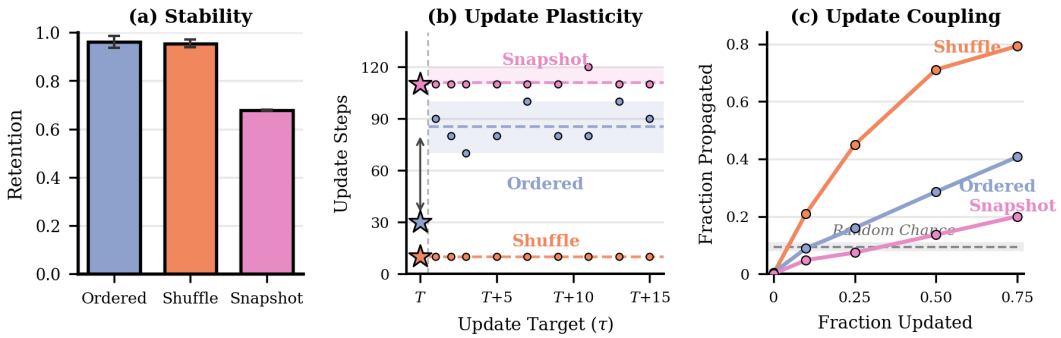


Figure 2: **Temporal Exposure Shapes Responses to Future Updates. (a) Stability.** We measure the accuracy on constant facts after training on the held-out $\tau = T$ time-step until 95% of fact updates are learned. Models exposed to multiple time-steps (SHUFFLE and ORDERED) preserve constant facts under the update; SNAPSHOT does not. **(b) Update Plasticity.** Gradient steps to reach 95% recall on a held-out update, versus the future time-step the update targets ($\tau = T$ is the anticipated successor step; larger values of τ advance each fact further into the future). ORDERED learns the successor update quickly ($\sim$30 steps) but is slow for targets further in the future (blue band), while SHUFFLE learns any future time-step's facts quickly ($\sim$10 steps). SNAPSHOT learns all time-steps slowly ($\sim$100 steps), serving as a control model that has not observed change. **(c) Update Coupling.** We update a random subset of facts to their $\tau = T$ answer and measure the fraction of untrained facts that move to their $\tau = T$ answer. Updates spread the most in SHUFFLE by a large margin, and less in ORDERED and SNAPSHOT.

further into the future. SHUFFLE is fast at updating to any future time-step. In Figure 7 (Appendix A), we compare the update speed of each model on the expected $\tau = T$ update with a purely random update for both SHUFFLE and ORDERED. Both models learn random updates significantly (5–10$\times$) slower than the $\tau = T$ update, providing further evidence that the fast updates observed here reflect a learned adaptation to how facts change in the data.

Update coupling. We update a random subset of shifting facts to their $\tau = T$ state and measure what fraction of untrained facts are also updated. We find that updates spread significantly more in SHUFFLE than in ORDERED, reaching about 2.5$\times$ as many untrained facts when updating half of the shifting facts. The SNAPSHOT model, which sees only one time-step during pretraining, spreads updates the least, indicating that this update coupling is induced by observing how facts change across multiple time-steps. SNAPSHOT's coupling above chance reflects answer sharing: facts that share an answer at the last training time-step also share their next answer, so an update can spread between them without any observed change. This answer sharing is identical for all three models, so it cannot explain why SHUFFLE couples more.

What kind of change produces coupling? In our setting, facts that change together also follow a shared ordering of answers. We separate the two in a smaller setting where groups of facts jump together to random answers, and each group shares its current answer with another group that moves elsewhere, so that neither an ordering nor a shared answer predicts where a fact moves (Figure 3(a), Appendix E). SHUFFLE still carries an update to 40% of untrained facts, compared to 28% for ORDERED and 21% for SNAPSHOT. This coupling cannot come from answer sharing, so co-movement alone is enough for SHUFFLE to group facts. The shared ordering strengthens this grouping (SHUFFLE 52%, ORDERED 39%, SNAPSHOT 19%), while corrupting individual facts' transitions, so that they no longer move with their group, removes it (Figure 3(b)). The grouping is robust to moderate noise: with 20% of transitions corrupted, SHUFFLE still couples more than SNAPSHOT (0.26 vs. 0.19), and it falls to SNAPSHOT's level only at $\eta \approx 0.35$.

Together, these probes indicate that observing changing knowledge shapes how models respond to future updates. The presentation of the data changes this response: ORDERED is preferentially plastic to the next time-step, while SHUFFLE learns any future time-step quickly and spreads updates to facts that are not trained on. In the next section, we study the mechanisms behind these differences.

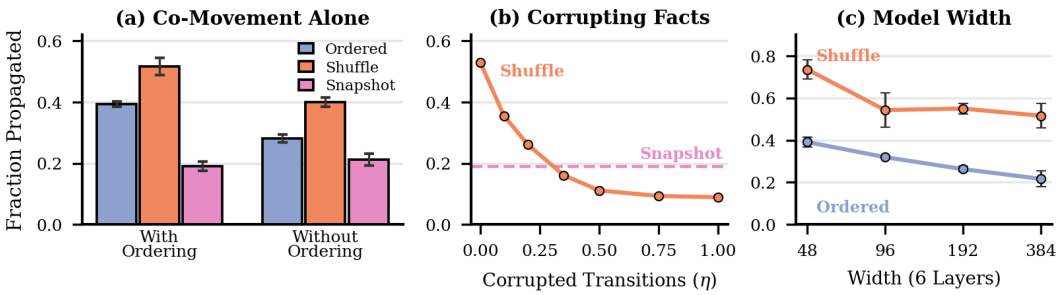


Figure 3: **What Produces Update Coupling.** Fraction of untrained facts that move to their $\tau = T$ answer after updating half of the shifting facts, in a smaller version of our setting. **(a) Co-Movement Alone.** With the shared ordering of answers and without it, where groups of facts transition together to random answers and a shared current answer does not predict a fact's next answer. **(b) Corrupting Facts.** SHUFFLE's coupling as a fraction η of individual facts' training transitions is replaced with random answers. **(c) Model Width.** Coupling from 0.18M to 10.8M parameters, trained for 24,000 steps so that all sizes fit.

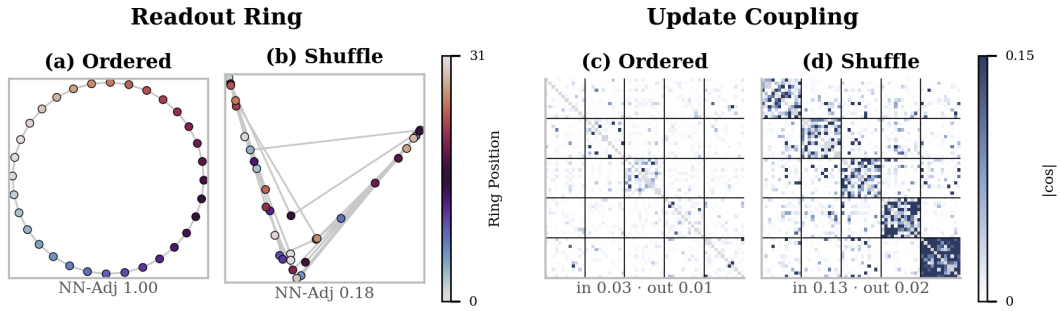


Figure 4: **Individual Slots versus Co-Movement Classes. (a, b) Token Geometry.** We show a Laplacian eigenmap of the answer-token embeddings, colored by position in the temporal progression. The ORDERED model embeds answer tokens such that temporally adjacent answers are nearest neighbors, presenting as a ring under the eigenmap transformation. SHUFFLE has no such geometry in the answer-token embeddings. **(c, d) Update Coupling.** We show the pairwise $|\cos|$ between the held-out-update gradients of every pair of facts, grouped by facts that start at the same offset in the answer progression. ORDERED is near-structureless, suggesting each fact occupies a disentangled slot in the model; SHUFFLE shows a block-wise similarity structure indicating that facts that move together share an update direction.

4.2 TWO FORMS OF FACT STORAGE: INDIVIDUAL SLOTS VS. CO-MOVEMENT CLASSES

Previously, we saw that ORDERED and SHUFFLE take on updates in different ways: ORDERED updates quickly but only to the expected next time-period answer, and its updates stay local, while SHUFFLE absorbs any update that preserves the co-movement structure and lets it spread to related facts. In this section, we find that these differences come from two different ways of storing facts.

Embedding geometry. Figure 4 (a) and (b) show a Laplacian eigenmap of the answer-token embeddings for the SHUFFLE and ORDERED models, with each point colored by its order in the answer ring. This analysis reveals a clear distinction between the SHUFFLE and ORDERED models. ORDERED arranges its answer tokens such that answers that follow each other on the ring are adjacent in embedding space. On the other hand, SHUFFLE shows no recognizable structure in the answer token geometry. We quantify this by measuring the fraction of answer tokens whose nearest neighbor in embedding space is one of their temporal neighbors (NN-Adj). For ORDERED models, this occurs in 100% of the cases, whereas for SHUFFLE this only occurs in 18% of the cases. Our results suggest that ORDERED encodes the temporal structure in the answer embedding space, arranging it so that advancing to the next time-step requires minimal parameter movement.

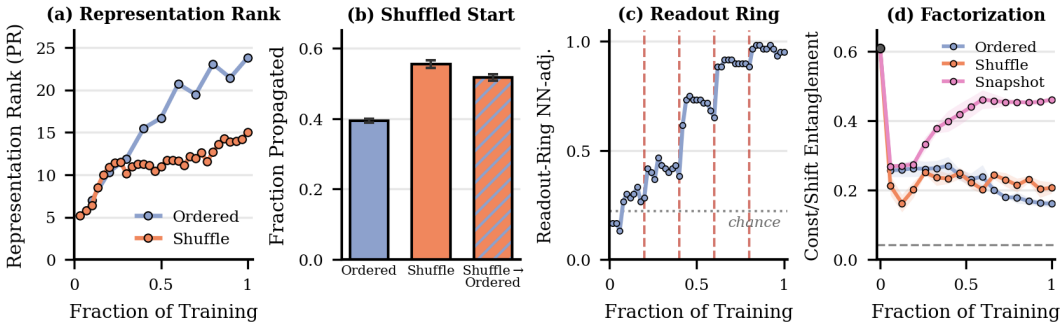


Figure 5: **Analyzing the Training Dynamics of ORDERED and SHUFFLE. (a) Representation Rank.** We measure the effective rank (participation ratio) of the shifting-fact activations throughout training. ORDERED's rank expands throughout training, while SHUFFLE's stays suppressed. **(b) Shuffled Start.** Update coupling at the end of training for ORDERED, SHUFFLE, and a model trained on shuffled time-steps for the first half of training and in chronological order for the second (Shuffle→Ordered), in a smaller version of our setting. **(c) Dynamics of Embedding Geometry.** We plot NN-Adj. of ORDERED's answer-token readout embeddings across training. The temporal adjacency structure of the answer-token embeddings forms in phase transitions as the model encounters new time-steps. **(d) Stability Dynamics.** We measure the cosine similarity of shifting and constant fact activations throughout training. SHUFFLE decouples the representations early in training and preserves this as it memorizes. ORDERED decouples shifting and constant facts as it encounters more time-steps transitions.

Update geometry. Turning from the answer tokens to the updates themselves, we ask how the held-out update acts on each fact. For every pair of shifting facts we compute the cosine similarity of their held-out-update gradients, grouping facts that start at the same position in the answer progression (and hence move together; Figure 4(c) and (d)). The two models again differ sharply. For ORDERED (Figure 4(c)), the update gradients are close to orthogonal and show no relational structure, so each fact is updated along its own direction. For SHUFFLE (Figure 4(d)), the gradients form clear within-relation blocks: facts that move together across time are updated along a common direction. This mirrors the update-coupling result of Figure 2(c): because SHUFFLE updates facts that moved together during training through a shared direction, updating some of them carries to the rest, whereas ORDERED's more independent update directions keep updates relatively local.

Our results point to two ways that models store facts and adapt to temporal structure, set primarily by how the data is presented. ORDERED stores facts in **individual slots**, updating each fact along its own direction, and captures the temporal structure in the geometry of the answer tokens. SHUFFLE instead stores facts that have moved together in **co-movement classes**, which share a low-rank update subspace. The distinction is one of degree: ORDERED still couples updates more than SNAPSHOT, and removing SHUFFLE's shared subspace lowers its coupling to ORDERED's level (Appendix B). The difference holds from 0.2M to 11M parameters and does not shrink as capacity grows (Figure 3(c), Appendix G), so it is not driven by capacity pressure.

4.3 THE ORIGINS OF TWO FORMS OF STORAGE

We next ask why presentation order determines how facts are stored, by studying the training dynamics of SHUFFLE and ORDERED. We find that the two models learn in opposite orders: SHUFFLE learns temporal structure early, before it memorizes the individual facts (Appendix C), while ORDERED accrues it gradually as it encounters successive time-steps.

Why does presentation change storage? We trace storage differences to training dynamics early in training. ORDERED sees consistent facts from a single time-step and can memorize them using relatively decoupled representations inherited from random initialization. In contrast, conflicting temporal knowledge initially prevents SHUFFLE from memorizing shifting facts, so memorization occurs only after shared representations have formed. This early divergence shapes the rest of training: SHUFFLE remains low-rank throughout, while ORDERED's rank keeps increasing (Figure 5(a)). If ORDERED's individual slots rely on the decoupled representations of random initialization, chrono-

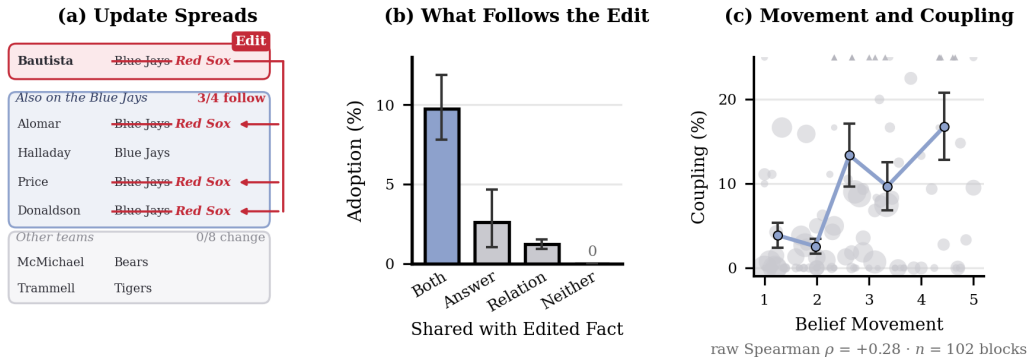


Figure 6: **Updates Spread Through Classes of Related Facts in OLMo-2-7B. (a) Update Spreads.** An example fine-tuning edit that changes José Bautista's team from the Toronto Blue Jays to the Boston Red Sox. Three of the four other players that the model places on the Blue Jays also move to the Red Sox, while players on other teams do not (0/8). This edit spreads more than the average edit in (b). **(b) What Follows the Edit.** The fraction of other facts that adopt the new answer across 947 edits, grouped by what they share with the edited fact. Error bars are 95% bootstrap intervals over edits. **(c) Movement and Coupling.** For each class (facts sharing a relation and a pre-edit answer), we plot coupling (the fraction of other members that adopt an edit to one member) against how much the model's answers for the class changed across pretraining checkpoints.

logical training that starts from a partially trained SHUFFLE model should not produce it. Figure 5(b) tests this: training on shuffled time-steps for the first half of training and in chronological order for the second yields update coupling close to SHUFFLE's (0.52 vs. 0.55) rather than ORDERED's (0.39).

How does ORDERED adapt to temporal structure? We next ask how ORDERED's per-fact representations adapt to temporal structure, protecting constant facts while remaining plastic to chronologically consistent updates. Figure 5(c) tracks the alignment between the answer-token ring and the chronological order of answers: this alignment rises sharply at each transition from time-step τ to $\tau + 1$, then degrades mildly within each time-step. Similarly, the entanglement between constant and shifting facts decreases only later in training (Figure 5(d)). Together, these results suggest that ORDERED first allocates individual, fact-specific slots and subsequently learns to rewrite these slots as it experiences transitions.

What drives co-movement classes? Our account attributes these classes to interference between a fact's conflicting versions. Removing the conflicting components between the per-time-step gradients within each batch with PCGrad (Yu et al., 2020) leaves SHUFFLE's rank and update coupling unchanged (Figure 10, Appendix D), so interference within a batch is not required. Interference across batches is sufficient: training in chronological order while cycling through time-steps more quickly, so that a fact's versions never share a batch but follow one another closely, produces co-movement classes, raising update coupling from 0.39 to 0.65 as phases shorten from 1,200 to 240 steps with the same facts, and hence the same answer sharing, (Appendix F). The same trend holds when training with plain SGD without momentum, so this interaction is carried by the weights rather than by the optimizer's state. Co-movement classes therefore form when successive updates present conflicting versions of a fact, whether or not these versions appear in the same batch.

5 CLASS-BASED STORAGE IN A PRETRAINED LANGUAGE MODEL

We next ask whether class-based storage appears in a pretrained language model. We study OLMo-2-7B (OLMo et al., 2025), whose pretraining corpus and intermediate checkpoints are public. We collect 6,136 facts about popular Wikidata entities across 20 relations, keeping facts that the model answers consistently under at least two of three phrasings. We then edit (update) one fact at a time with full fine-tuning, stopping once the model produces the new answer under a held-out phrasing. We define classes here as groups of facts that share the same relation and answer.

Updates spread within a class. In Figure 6(a), we show an edit that moves José Bautista from the Toronto Blue Jays to the Boston Red Sox. This edit also moves three other Blue Jays players (Alomar, Price, and Donaldson) to the Red Sox, while players on other teams are unaffected. Across 947 edits (Figure 6(b)), edits spread primarily to facts that share both the relation and the answer of the edited fact, 9.8% of which adopt the new answer. Facts that share only the answer (2.6%) or only the relation (1.2%) adopt it far less often, and unrelated facts never do. A shared answer alone therefore explains little of the spread. As in SHUFFLE, an update spreads to part of its class (facts with the same relation and answer).

Where do classes come from? In our controlled setting, co-movement classes form among facts that change together during training. As an exploratory test in OLMo, we measure each class's movement, the number of distinct answers the model gives for its members across five pretraining checkpoints. Since all members of a class share the same pre-edit answer, and coupling only counts adoption within a class, this comparison controls for answer sharing. We use the model's own answers because Wikidata's edit history mostly reflects how thoroughly editors documented an entity (only 5.4% of "owned by" facts have more than one recorded value). We find that classes whose answers moved more are more strongly coupled when edited (Figure 6(c); Spearman $\rho = 0.28$, $n = 102$ classes). Controlling for the model's predictive entropy leaves it essentially unchanged (partial $\rho = 0.27$), but the result is correlational (due to the infeasibility of intervening at pretraining scale).

6 DISCUSSION

In this work, we have studied what models learn from observing changing knowledge. We show exposure to changing knowledge shapes how models respond to future updates, and models store knowledge differently depending on how they encounter it: chronological order induces storage in **individual slots**, which keeps updates local, while shuffling induces storage in **co-movement classes**. We now discuss the implications of these findings.

Implications for fact storage. Current analyses of fact storage in transformers take a static viewpoint, characterizing it as an associative lookup table (Geva et al., 2021; Nichani et al., 2024; Dugan et al., 2025), or, more recently, as geometric structure supporting implicit reasoning (Noroozizadeh et al., 2026; Nishi et al., 2025). In this work, we instead view fact storage mechanisms through their impact on downstream updates and demonstrate how pretraining dynamics trade off between coherent generalization of updates and specificity.

Implications for model updating. Keeping models up to date requires integrating new information while mitigating forgetting. The prevailing approach designs the update itself, through regularization, selective parameter updates, and data engineering. Our results suggest an alternative: pretraining can prepare models so that downstream updates are more effective, potentially reducing the need for replay data and extensive hyperparameter tuning.

Out-of-context reasoning and hallucinations. Storage in co-movement classes is a form of out-of-context reasoning, which can surface as hallucination when facts stop moving together (Huang et al., 2026). Prior work attributes this trade-off to the implicit bias of gradient descent. Here, we show that both forms of storage arise under the same model and objective, and differ only in data presentation.

Limitations and outlook. Our controlled setting lets us attribute the effect to temporal presentation, and Section 5 finds class-based storage in a pretrained language model. However, we cannot vary how a pretrained model observed its data, so whether these classes causally arise from temporal change, as in our controlled setting, remains open. Future work can identify techniques beyond data presentation that shape how knowledge is stored.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and cleaning and reformatting datasets. We have not used generative AI tools for interpreting results, for designing or providing feedback on research methodology or experiments, or for generating synthetic datasets. Developing theoretical models or conceptual frameworks, formulating mathematical claims, and proving them are not applicable to this work. Additionally, we used generative AI tools for generating plots, launching experiments at the request of the author, and helping draft initial sections of the text. We have

reviewed all AI-assisted work. Authors have verified the output of generated code, the accuracy of plots, and edited/corrected generated text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Afra Feyza Akyürek, Ekin Akyürek, Leshem Choshen, Derry Wijaya, and Jacob Andreas. Deductive closure training of language models for coherence, accuracy, and updatability, 2024. URL https://arxiv.org/abs/2401.08574.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less, 2024. URL https://arxiv.org/abs/2405.09673.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models, 2023. URL https://arxiv.org/abs/2307.12976.

Owen Dugan, Roberto Garcia, Ronny Junkins, Jerry Liu, Dylan Zinsley, Sabri Eyuboglu, Atri Rudra, and Chris Ré. Constructing efficient fact-storing mlps for transformers, 2025. URL https://arxiv.org/abs/2512.00207.

Lawrence Feng, Gaurav R. Ghosal, Jacob Mitchell Springer, Ziqian Zhong, and Aditi Raghunathan. Early data exposure improves robustness to subsequent fine-tuning, 2026. URL https://arxiv.org/abs/2605.12705.

Roberto Garcia, Jerry Liu, Ronny Junkins, Sabri Eyuboglu, Atri Rudra, and Christopher Ré. Mlps are hebbians: Constructing efficient fact-storing mlps for transformers, 2026. URL https://arxiv.org/abs/2607.10034.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. URL https://arxiv.org/abs/2012.14913.

Gaurav Ghosal, Tatsunori Hashimoto, and Aditi Raghunathan. Understanding finetuning for factual knowledge extraction, 2024. URL https://arxiv.org/abs/2406.14785.

Yixiao Huang, Hanlin Zhu, Tianyu Guo, Jiantao Jiao, Somayeh Sojoudi, Michael I. Jordan, Stuart Russell, and Song Mei. Generalization or hallucination? understanding out-of-context reasoning in transformers, 2026. URL https://arxiv.org/abs/2506.10887.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Emmy Liu, Kaiser Sun, Millicent Li, Isabelle Lee, Lindia Tjuatja, Jen tse Huang, and Graham Neubig. What do language models learn and when? the implicit curriculum hypothesis, 2026. URL https://arxiv.org/abs/2604.08510.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.08840.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower (ed.), *Psychology of Learning and Motivation*, volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

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. URL https://arxiv.org/abs/2202.05262.

Eshaan Nichani, Jason D. Lee, and Alberto Bietti. Understanding factual recall in transformers via associative memories, 2024. URL https://arxiv.org/abs/2412.06538.

Kento Nishi, Rahul Ramesh, Maya Okawa, Mikail Khona, Hidenori Tanaka, and Ekdeep Singh Lubana. Representation shattering in transformers: A synthetic study with knowledge editing, 2025. URL https://arxiv.org/abs/2410.17194.

Shahriar Noroozizadeh, Vaishnavh Nagarajan, Elan Rosenfeld, and Sanjiv Kumar. Deep sequence models tend to memorize geometrically; it is unclear why, 2026. URL https://arxiv.org/abs/2510.26745.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, 2019. URL https://arxiv.org/abs/1909.01066.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://arxiv.org/abs/1611.07725.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. URL https://arxiv.org/abs/2002.08910.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Jiaxin Wen, Zhengxuan Wu, Dawn Song, and Lijie Chen. Generalization dynamics of lm pre-training, May 2026. URL https://jiaxin-wen.github.io/blog/generalization-dynamics.html. Blog post.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2001.06782.

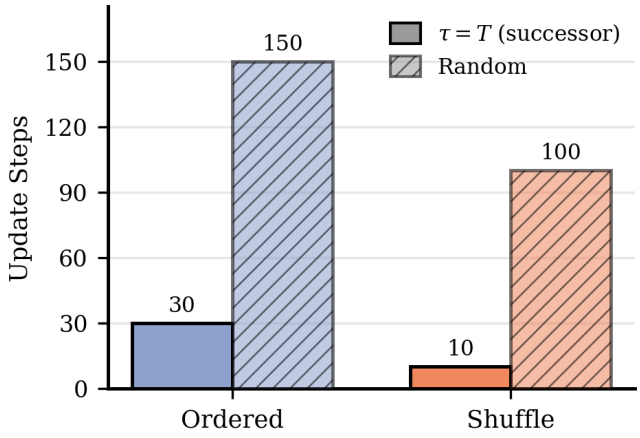


Figure 7: **Plasticity to Random Updates.** Gradient steps needed to reach 95% recall on the held-out update, for the $\tau = T$ update (solid) and for a random update that assigns each shifting fact an arbitrary answer from its answer set (hatched). Results are from the T=30 setting.

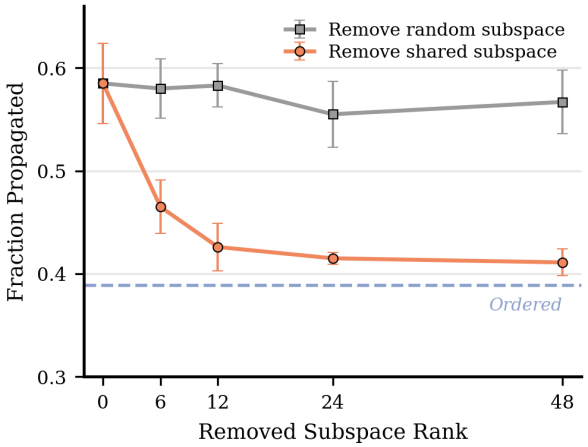


Figure 8: **Removing the Shared Subspace Lowers SHUFFLE's Coupling to ORDERED's Level.** Fraction of untrained facts that follow the held-out update when a subspace is projected out of every gradient step of the update, as a function of the subspace's rank. Projecting out SHUFFLE's shared subspace (orange) lowers coupling from 0.59 to about 0.41, close to ORDERED's 0.39 (dashed). Projecting out a random subspace of the same rank (gray) has little effect. The updated facts are still fully learned (recall 1.0). Error bars are over three probe runs on one trained model.

A PLASTICITY TO RANDOM UPDATES

In the main text, we found that SHUFFLE learns updates to any future time-step quickly (Figure 2(b)). Here, we test whether this reflects how facts changed during training or a general ease of learning new facts. We compare the $\tau = T$ update against a random update, in which each shifting fact is assigned an arbitrary answer from its answer set instead of the next answer in its sequence.

Figure 7 shows that both models learn the $\tau = T$ update quickly (30 steps for ORDERED and 10 for SHUFFLE), but take 5–10$\times$ longer to learn the random update (150 and 100 steps). This suggests that fast updating in both models is specific to updates that are consistent with how facts changed during training.

B Ablating the Shared Low-Rank Subspace

We test whether SHUFFLE's update coupling depends on the shared low-rank subspace. For each shifting relation, we collect the directions that move all of its facts to their updated answers , and call their span the shared subspace. During the held-out update, we project this subspace out of every gradient step and measure how many untrained facts still follow (Figure 8). Removing the shared subspace lowers SHUFFLE's coupling to roughly ORDERED's level, while removing a random subspace of the same rank has little effect. A subspace of rank 6–12 accounts for most of the drop, and larger subspaces do not lower coupling further. This suggests that the shared subspace accounts for the coupling that SHUFFLE adds beyond ORDERED, while the remaining coupling is shared by both models.

C Update Rank During Training

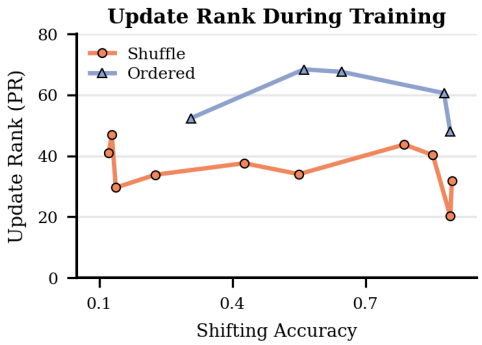


Figure 9: **SHUFFLE's Updates Are Low-Rank Before It Fits Shifting Facts.** Effective rank (participation ratio) of the per-fact training gradients on the last-block MLP output projection, as a function of shifting-fact accuracy during training. ORDERED is probed at its current time-step. Only accuracies below 0.95 are shown, since the gradients vanish at convergence.

In the main text, we argue that SHUFFLE learns shared temporal structure early, before it memorizes the individual facts. We test this by tracking the effective rank of the per-fact training gradients as a function of how well the model fits the shifting facts (Figure 9). A low rank means that facts are updated along a few shared directions, while a high rank means that each fact moves along its own direction. SHUFFLE's gradient rank is already low (about 30) while shifting accuracy is near 0.1, and stays between 30 and 45 as the facts are fit. ORDERED's gradients are higher-rank at every level of fit it reaches (about 50–70). This suggests that SHUFFLE's shared update directions form before the facts are memorized.

D Removing Within-Batch Gradient Conflict

When all time-steps are mixed, the versions of a shifting fact appear in the same batch with different answers, so their gradients conflict. One hypothesis is that co-movement classes arise from this conflict, since summing conflicting per-fact gradients could average them into a shared direction. If so, removing the conflict should lead to individual slots. We test this with PCGrad (Yu et al., 2020), which projects each per-time-step gradient onto the normal plane of any gradient it conflicts with, and with an Anti-PCGrad variant that removes the agreeing component instead.

Neither variant changes SHUFFLE's storage (Figure 10). Across nine runs, both remain low-rank and spread updates as much as standard SHUFFLE, well above ORDERED. Conflict between gradients in the same batch is therefore not required for co-movement classes. In Appendix F, we show that interference across batches is sufficient.

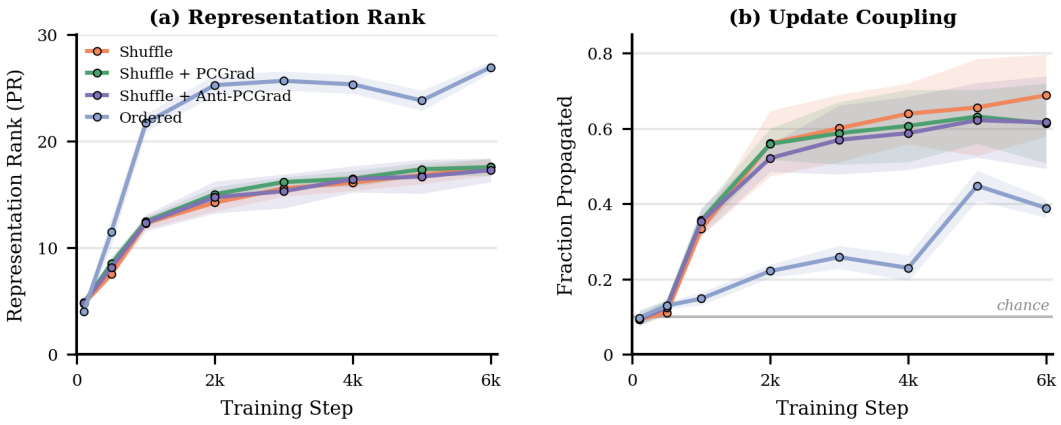


Figure 10: **PCGrad Does Not Change SHUFFLE's Storage. (a)** Representation rank and **(b)** update coupling during training, for standard SHUFFLE, SHUFFLE with PCGrad (removing the conflicting component between per-time-step gradients), SHUFFLE with Anti-PCGrad (removing the agreeing component instead), and ORDERED. Mean over nine runs, shaded ± 1 s.d.

E WHICH PATTERNS OF CHANGE PRODUCE UPDATE COUPLING?

We vary how shifting facts change across time-steps in the training data, and ask which properties of that change lead SHUFFLE to form co-movement classes. We separate two properties that coincide in our main setting: whether facts change together (co-movement), and whether their answers follow a shared ordering.

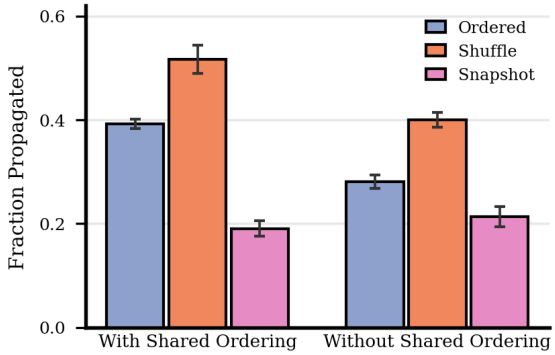


Figure 11: **Co-Movement Without a Shared Ordering.** Fraction of untrained facts that move to their $\tau = T$ answer after updating half of the shifting facts, in the smaller setting with the shared ordering (left) and without it (right). Facts co-move in both settings. Without the ordering, groups of facts jump together to new, randomly chosen answers, and pairs of groups share their current answer by coincidence. Mean $\pm$ standard error over three runs.

Co-movement without a shared ordering. In our main setting, facts that change together also follow a shared ordering of answers, so we cannot tell which of the two produces update coupling. We separate them in the smaller setting by removing the ordering while keeping co-movement. Each group of co-moving facts now jumps at every time-step to a new, randomly chosen answer shared by all of its members, and the held-out step is always such a jump. Because co-moving facts also share their current answer, a model trained on a single time-step could carry an update to other facts with the same answer without observing any change. To remove this shortcut, we pair the groups within each relation so that both groups in a pair hold the same answer at the last training time-step, but have different histories and move to different answers at the held-out step. Facts that share a current answer then share their next answer only 37% of the time, compared to 99% with the shared ordering.

Figure 11 shows that without the ordering, SHUFFLE carries the update to 40% of untrained facts, compared to 28% for ORDERED and 21% for SNAPSHOT. Since a fact's current answer no longer indicates its group, this coupling cannot come from answer sharing. SHUFFLE therefore groups facts by how they have moved together, even without a shared ordering, while ORDERED learns little of this structure. With the shared ordering, the gains from temporal exposure are larger (SHUFFLE 52%, ORDERED 39%, SNAPSHOT 19%), and SHUFFLE's representation is lower-rank (rank 11, compared to 20 without the ordering). SNAPSHOT stays above chance because only pairs of groups share an answer, so the current answer is still partially informative.

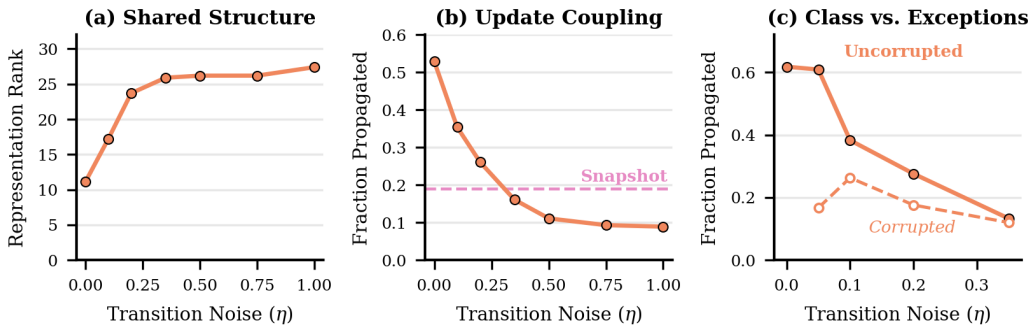


Figure 12: **Co-Movement Classes Require Facts to Change Together.** We replace a fraction η of shifting facts' training transitions with a random answer from the fact's own cycle, and evaluate on the uncorrupted data. **(a) Shared Structure.** SHUFFLE's representation rank rises steeply with η. **(b) Update Coupling.** Fraction of untrained facts that move to their $\tau = T$ answer after updating half of the shifting facts. SHUFFLE's coupling falls to the level of SNAPSHOT (dashed) by $\eta \approx 0.35$, and to near chance at $\eta = 1$. **(c) Class vs. Exceptions.** SHUFFLE's update coupling, measured separately for facts whose transitions were never corrupted and for corrupted facts.

Corrupting individual facts. We next test how SHUFFLE's storage responds when individual facts stop changing with their group. We use the smaller setting (100 entities, answer cycles of length 10, 6,000 training steps). We corrupt a fraction η of shifting facts' training transitions by replacing the answer with a random element of the fact's cycle. The corruption is fixed for the run, constant facts are unaffected, and we evaluate on the uncorrupted data. Unlike the random-jump setting above, where all members of a group make the same jump, the corruption is applied to each fact independently, so a corrupted fact no longer moves with its group. We measure update coupling as in the main text, by updating half of the shifting facts to their $\tau = T$ answer.

Breaking co-movement removes the shared structure. As η increases, SHUFFLE's representation rank rises from 11 without noise to 24 at $\eta = 0.2$ and 27 at $\eta = 1$ (Figure 12(a)). Update coupling falls with it, from 0.53 to SNAPSHOT's level (0.19) by $\eta \approx 0.35$, and to 0.09, near chance, at $\eta = 1$ (Figure 12(b)). Together with the random-jump result, this suggests that mixing time-steps produces co-movement classes when facts change consistently together, whether or not they follow a shared ordering. Coupling falls below SNAPSHOT at high noise because corrupted facts no longer move with any group and the model fits the uncorrupted data less well.

A few exceptions coexist with the class. We measure update coupling separately for facts whose transitions were never corrupted and for corrupted facts (Figure 12(c)). At $\eta = 0.05$, never-corrupted facts keep the noise-free level of coupling (0.61 vs. 0.62), while corrupted facts are stored as exceptions (0.17). At higher noise, coupling falls even among never-corrupted facts (0.38 at $\eta = 0.1$). This suggests that the class is stored as shared structure, since corrupting some of its members degrades coupling for all of them, including members that still move together.

F INTERFERENCE ACROSS BATCHES

PCGrad removes conflicting gradients within each batch but does not change SHUFFLE's storage (Appendix D). Here, we test whether interference can instead act across batches. In the smaller setting (100 entities, answer cycles of length 10, 6,000 training steps), we train in chronological order

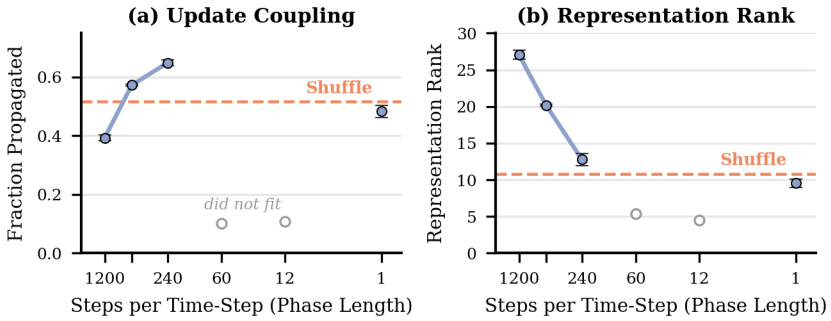


Figure 13: **Co-Movement Classes Form Without Within-Batch Conflict.** ORDERED trained by cycling through the time-steps in chronological order several times, with the number of steps spent on each time-step (the phase length) decreasing from 1,200 (standard ORDERED, one pass) to 1 (alternating time-steps between consecutive batches). Each batch contains a single time-step, so a fact's versions never appear in the same batch. **(a)** Update coupling and **(b)** representation rank. The dashed line shows SHUFFLE. Hollow points did not fit the training data (shifting accuracy below 0.3) and are excluded. Mean ± standard error over three runs.

but cycle through the time-steps $\tau = 0, \ldots, 4$ several times, spending a fixed number of steps on each (the phase length). Total training steps are held fixed, and every schedule ends on $\tau = 4$. Since each batch contains a single time-step, a fact's conflicting versions never share a batch. Shorter phases only make them follow each other more closely.

Shorter phases produce co-movement classes. Shortening the phase length from 1,200 steps (standard ORDERED) to 600 and 240 steps raises update coupling from 0.39 to 0.57 and 0.65, above SHUFFLE's 0.52 (Figure 13). Representation rank falls from 27 to 13. Alternating time-steps between consecutive batches (a phase length of 1) behaves like SHUFFLE (coupling 0.48, rank 10). Interference between a fact's versions therefore does not require them to share a batch.

The effect does not depend on the optimizer's state. Adam keeps running gradient statistics across steps, which could carry the interaction between batches. We therefore repeat the sweep with plain SGD without momentum (learning rate 0.1, 24,000 steps so that all schedules fit), scaling the phase lengths to keep the same number of passes through the time-steps. We see the same trend. Update coupling rises from 0.53 for standard ORDERED (one pass) to 0.67 with two passes and 0.89 with five, above SHUFFLE's 0.76, and representation rank falls from 24 to SHUFFLE's level (13–14). Since plain SGD keeps no state between steps, the interaction must be carried by the weights. Updates spread more in both models under SGD than under Adam, so values should only be compared within an optimizer.

G ROBUSTNESS TO MODEL SIZE

We test whether the difference between ORDERED and SHUFFLE depends on model size. In the smaller setting, we vary width from 48 to 384 at 6 layers and depth from 2 to 8 layers at width 192. We train every model for 24,000 steps so that all sizes fit the training data (shifting accuracy ≥ 0.98). Updates spread more in SHUFFLE than in ORDERED at every size, by 0.22 to 0.44 (Figure 14), and the gap does not close at the largest width (0.52 vs. 0.22 at 10.8M parameters). ORDERED's coupling is lower than in the main text because of the longer training budget, so values should be compared within a model size. SHUFFLE's representation rank is lower than ORDERED's at every size except the largest width, where the two are similar. We therefore use update coupling as the main measure across sizes.

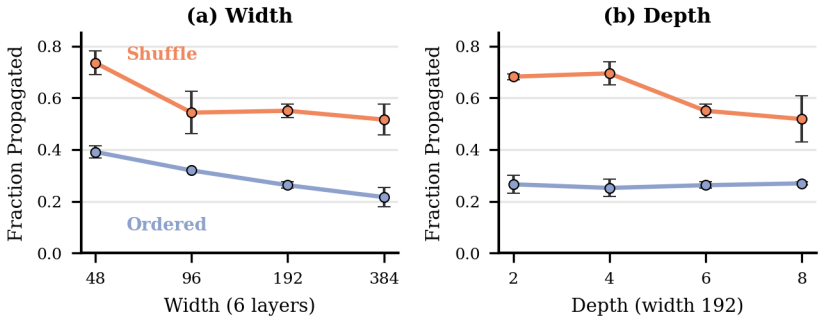


Figure 14: **Update Coupling Across Model Sizes.** Fraction of untrained facts that move to their $\tau = T$ answer after updating half of the shifting facts, for ORDERED and SHUFFLE. We vary **(a)** width from 48 to 384 at 6 layers (0.18M to 10.8M parameters) and **(b)** depth from 2 to 8 layers at width 192. All models fit the training data. Mean $\pm$ standard error over two runs.