COMPARTMENTALIZATION: LEARNING TO TRANSLATE IS NOT LEARNING TO TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models learn from data presented in many surface forms; they train on the same facts in English and Swahili, the same functions in Python and Haskell. Ideally, what a model learns from one form should help it with others. We test a variant of this in a controlled setting by rendering the same text in c *compartments* that use disjoint vocabularies, each a token-for-token relabeling of the others. Models up to 42M parameters trained on 131B tokens often learn each compartment about as slowly as if trained on its share of the data alone, a failure we call *compartmentalization*. Models initialized with identical embeddings across compartments show no such cost, so a unified solution exists, but is not found from generic initialization. Paired examples, each showing the same text in two compartments, teach models to convert between compartments almost perfectly (99.7–100% exact match), but this reduces the sample efficiency cost only past a threshold determined by c and the learning rate. Learning a mapping between surface forms, in other words, does not trivially imply transfer learning between them.

1 INTRODUCTION

Language models learn from data abundant with distinct surface forms of unified concepts. A single fact may appear in English, then Russian, then Mandarin; a code function might be expressed in both Python and Haskell; a proposition can be expressed in both formal and natural language. Ideally, a model would recognize these distinct forms as related. Indeed, much of the success of large language models suggests that they frequently do, but there is growing evidence that frontier models still often fail to exhibit robust transfer across different surface forms (§2).

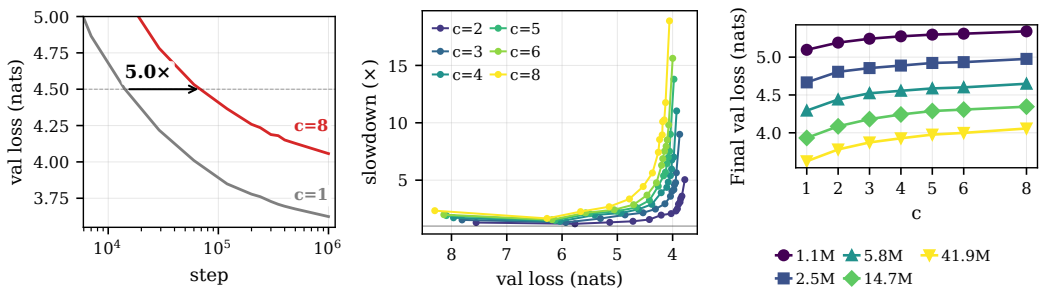
We invoke the notion of an *interlingua* from machine translation: a semantic space shared by multiple surface forms. Johnson et al. (2017) showed that a shared encoder can sometimes learn such a space, translating between language pairs it was never trained on. We study an instance where a model fails to do this, a failure mode which we call *compartmentalization*. A model's learning process is compartmentalized to the degree that each surface form is learned no faster than if the model had been trained on that form's share of the data alone. In the worst case, compartmentalization is total redundancy: the same structure, learned independently for each form and stored as separate representations, with no amortization between them.

We focus on two consequences of compartmentalization:

- **Sample inefficiency.** If every latently similar surface form is treated as a separate prediction task, each benefits only from its own share of the data. In the worst case, where one such form accounts for only ε of the data, it could take $1/\varepsilon$ as many data samples to learn a similar amount of structure under that form.
- **Capacity competition.** Redundant representations also consume finite representational capacity. Related multilingual work (Conneau et al., 2020) observes that models trained on many languages fail to effectively share fixed model capacity.

We study a worst case of compartmentalization, in which the same text is encoded in c surface forms with disjoint vocabularies and no correspondence between forms is given a priori (§3). Our contributions are as follows:

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.



(a) A demonstration of slowdown. (b) Sample efficiency cost. (c) Capacity cost.

Figure 1: **Compartmentalization costs sample efficiency and capacity.** a To calculate slowdown: at any val-loss target L achieved by some checkpoint of the base c=1 model (here L=4.5), find the steps where c=1 and c=N each cross L, and take their ratio. We linearly interpolate between nearby checkpoints as needed. The example reads: a 5.0× slowdown for c=8 at L=4.5. b Slowdown of each c-compartment run vs. the c=1 baseline on the 41.9M model, plotted against the matched val-loss target. Slowdown grows with c across the trajectory. c Final val loss vs. c, across five model scales (1.1M to 41.9M parameters). Final val loss also grows with c.

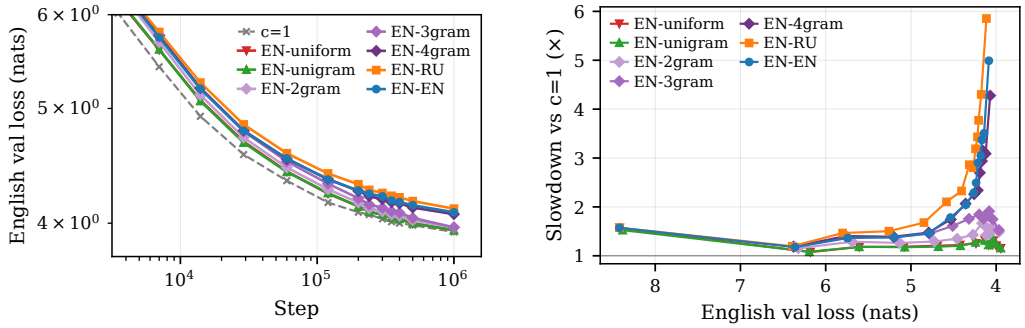
- **Costs of compartmentalization.** We observe a sample efficiency slowdown and a validation-loss gap that both increase with c, across models from 1.1M to 41.9M parameters; this gap does not meaningfully improve with continued training (§4).
- **Existence of unified solutions.** Models whose token embeddings are initialized as copies across sub-vocabularies, or duplicated from a trained c=1 model, match the c=1 baseline. We demonstrate that solutions without a loss gap exist but standard training from a generic initialization does not find them (§4.1).
- **Paired data is often insufficient.** Models trained with paired "translation" examples learn the demonstrated mapping between forms almost perfectly, but this data does not improve learning transfer until a threshold determined jointly by c, the fraction of overall data represented as "translation" pairs tr, and the learning rate L. At c=2, we do not observe meaningful reduction at any learning rate we test. Weight decay accelerates the threshold to smaller values of c and tr (§4.3).
- **Bilingual transfer.** Using models trained on English and Chinese as a case study, our construction shows that a shared tokenizer barely improves over disjoint vocabularies (§4.4).

2 Related Work

Multilingual pretraining provides the closest analog to our construction. Conneau et al. (2020) document a *curse of multilinguality*: at fixed capacity, adding languages helps lower-resource languages to a point before eventually degrading performance. Huang et al. (2026) trace a related failure to neuron competition: small models allocate capacity to frequent or simple tasks over rare tasks even when solutions for rare tasks exist, while larger models handle both types of task due to lower gradient interference.

Ifergan et al. (2025) show that 7B-scale models answer correctly in at least one language roughly three times as many facts as they answer correctly across the 13-language average—and about 150% more than in their best single language—evidence that factual knowledge is siloed across languages. Similarly, Goldman et al. (2025) demonstrate that non-thinking frontier language models consistently fail to recall in other languages Wikipedia facts that exist only in a single language. Qi et al. (2023) find that scale raises per-language factual accuracy without improving cross-lingual consistency, paralleling Ifergan et al. (2025).

Within a single language, Calderon et al. (2026) find that frontier language models can correctly recite many facts in a pretraining-like Wikipedia context, but not in a QA-like context. In these cases, information is encoded in brittle ways that prevent retrieval—the model does not functionally "know" these facts. In a similar vein, Berglund et al. (2024) introduce the Reversal Curse, one of the more



(a) English val loss over training.

(b) Slowdown vs. the single-compartment baseline.

Figure 2: **English validation loss gap increases with the complexity of data in the other compartment.** For 14.7M base model with c=2, we compare seven choices for the second compartment's data—English (a control), Russian, text sampled from n-gram models of order $n \in \{2, 3, 4\}$ estimated on 1B tokens of the same corpus (see Appendix I), unigram-frequency-sampled noise, and uniform-random-token noise—vs. the single-compartment (c=1) baseline. All values are averages over three seeds. a English-side validation loss for each condition. b Slowdown of each condition vs. the c=1 baseline. Slowdown increases with complexity of the other dataset.

well-known and well-replicated frontier failures, wherein LLMs that see factual relations of the form "A is B" during training fail to generalize to the implied but unseen form "B is A".

Wendler et al. (2024) find that Llama-2's concept space lies closer to English than to other languages, and Dumas et al. (2025) show via activation patching that concept and output language can be varied independently, evidence of language-agnostic concept representations.

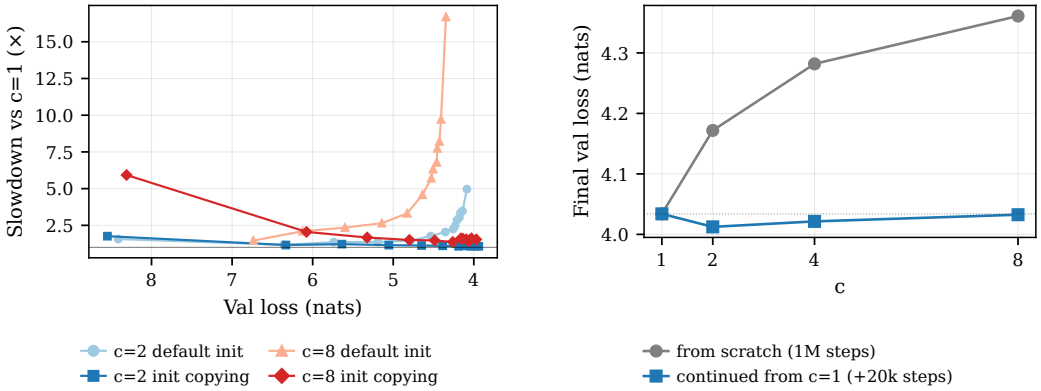
The generalization threshold we observe resembles delayed generalization in small networks. In *grokking*, a generalizing solution exists and is eventually found, but only after the model first fits a memorizing one, and weight decay strongly controls when the switch happens (Power et al., 2022; Nanda et al., 2023; Liu et al., 2023).

3 Methodology

To systematically explore compartmentalization, we first describe a method to create c distinct surface forms from the same data source. Because data from these compartments are statistically identical, in the best case, models could theoretically share a tremendous amount of representational and computational structure across them, but in the worst case, they would waste modeling capacity by treating them as c independent tasks. Our method is designed to induce compartmentalization in isolation from the structural cues that encourage representation sharing in natural data (subword overlap, syntactic regularity, semantic correspondence). Similar techniques have been employed by works like K et al. (2020) and Schäfer et al. (2024) to study cross-lingual generalization.

We create a maximally compartmentalized model with c surface forms with a simple tokenizer trick: increasing the base size of the embedding table V by c times to obtain a vocabulary with cV tokens. Then, to encode data in surface form j, we simply offset token IDs by Vj. Although this allows us to encode arbitrary ratios of data in each surface form, we assume an equal c-way split in our experiments. In this construction, forms equally share the negative effects of compartmentalization. Throughout this work, we refer to these forms as *compartments*. We initialize the parameters in the expanded embedding table in the normal way, without any a priori correspondence between tokens i and $i + Vj$. §4.4 applies this construction to a pair of natural languages to measure how much the subword overlap contributes.

Training setup We address both sample efficiency and capacity by training multiple scales of small models on a fixed token budget of ∼160–5000× the Chinchilla-optimal count (Hoffmann et al., 2022) for their scale. Specifically, we train GPT-style decoders (§A.1) on ∼131 billion tokens of FineWeb



(a) Slowdown for init-copied models vs. c=1 baseline.

(b) Post-hoc parameter duplication finetuning.

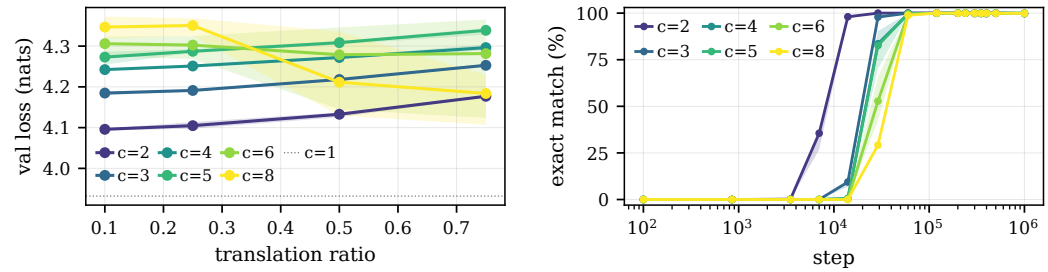
Figure 3: **Validation curves for unified solution experiments.** a Slowdown of compartmentalized and init-copied runs (vs. c=1 baseline) at matched val loss, for c=2 and c=8, at 14.7M scale. Runs initialized by copying embedding rows from first compartment to every other compartment track c=1 throughout training. b Minimal continued pretraining (20,000 steps, or 2% of previous total) on a c=1 model allows compartment-specific performance matching the c=1 baseline, i.e. without a capacity cost.

(Penedo et al., 2024), i.e., 1M steps with a batch size of 2048, a block size of 64, and model sizes (see Table 1) we can train on a single GPU due to compute constraints. We train all experiments with a fixed learning rate of 2×10^{-5} and no weight decay, which we found had limited effect on fully-compartmentalized models (§4.3.1); §4.3 shows how a larger learning rate changes our paired-data results. We accept the additional limitation of using a 16k tokenizer we train ourselves, breaking any perplexity comparisons with established models, to mitigate the memory cost of multiplying the vocabulary size up to eight times.

Measuring sample inefficiency We compare every compartmentalized model to a more typical control: a compartment-free c=1 baseline trained on the same underlying tokens, in the same order, with the same hyperparameters. We report the difference in averaged per-compartment validation loss between each condition and this baseline, and *slowdown*: a ratio of steps taken, between compartmentalized and c=1 models, to reach a given validation loss (Fig. 1a).

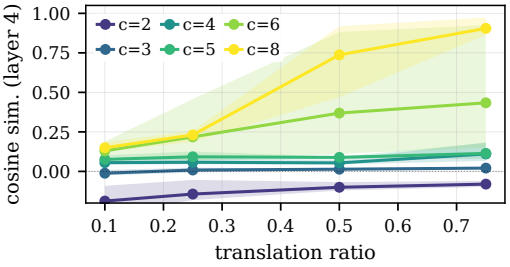
Evaluating representational similarity To test whether a model represents the same text similarly in different compartments, we encode a fixed batch of 64 held-out FineWeb sequences and record the hidden state at a middle-layer output (layer 4). For each pair of compartments, we compute the cosine similarity between the hidden states at each token position, then average over positions and over all pairs of compartments (§D.3). A value near 1 means the model assigns the same text nearly the same hidden state in every compartment.

Cosine similarity captures whether two compartments' activations point in the same direction, but not how functionally compatible they are. We also test representational similarity from a causal perspective using activation patching (Vig et al., 2020; Geiger et al., 2021; Meng et al., 2022) across compartments and measuring the effect on the model's prediction. We designate a 'host' compartment, in which the prompt prefix is computed up to an intermediate layer, and a 'donor' compartment, from which we take the final-token activation to be patched. We patch the donor activation at the end of the host prefix, complete the forward pass to obtain a next-token distribution, and measure the Jensen-Shannon divergence between the patched and original distributions. Lower JSD indicates higher cross-compartment functional compatibility at that activation site. We perform this in both directions and average the JSDs, then report the median across 2,000 FineWeb prompts.

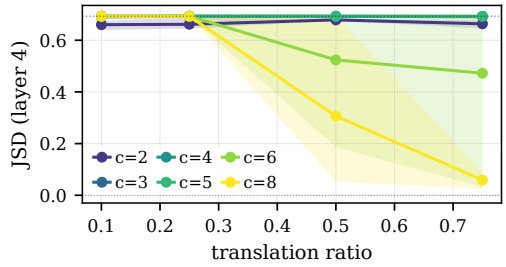


(a) Final val loss vs paired data ratio.

(b) Pair-completion acc. (tr=0.5).



(c) Cross-compartment cossim.



(d) Cross-compartment patch JSD.

Figure 4: **For the 14.7M param configuration, at our learning rate, paired data has a slightly negative effect until a phase transition at c=8. The pair-completion task itself is learned at every c, but below the transition, this does not serve to reduce compartmentalization.** We plot each c over 3 seeds. a Final per-compartment validation loss vs. paired data ratio, one line per compartment count c. Dotted line is the c=1 baseline. b Pair-completion accuracy: the second half of each held-out paired example is greedily decoded from the first half and scored by exact matches, averaged across pairs of compartments. Models learn the pair-completion task quickly: every condition reaches 99% by step 1.2×10^5. c Mean per-token cosine similarity between middle-layer activations of text encoded in different compartments, vs. paired data ratio. d Jensen-Shannon divergence between next-token distribution with and without patching in the layer-4 activation from another compartment for the final token (median over 2,000 prompts, averaged over both patch directions).

4 Experiments

The results of our experiments are in Figure 1. **Supporting our characterization of compartmentalization, we observe *sample efficiency* slowdown (vs. c=1) increasing with c, and a validation-loss gap (also vs. c=1) that widens with c and does not meaningfully shrink under continued overtraining, indicative of *capacity competition*.** Note that the c=8 model has strictly more *total* parameters than the c=1 baseline (Table 1) and nonetheless performs worse: total-parameter differences therefore cannot explain the observed effect. Reinforcing this, a parameter-matched control—the same enlarged vocab embedding, but trained like c=1—does not exhibit a similar cost (Fig. 12). Across checkpoints, downstream accuracy on three benchmarks is predicted by validation loss alone, regardless of c or tr (Appendix F).

4.1 Existence of unified solutions

Results in the previous section raise a natural question: is it even possible for our models to share strength across compartments? We test this in two ways: initialization duplication and post-hoc parameter duplication.

4.1.1 Initialization duplication

In this condition, instead of initializing all tokens independently, we initialize only the first V tokens corresponding to the base vocabulary, and copy their values to every other range $Vj \dots V(j+1)$, without aliasing the ranges to each other. For example, at initialization, the embedding for "apple" is identical across all compartments, but these parameters are allowed to diverge during training.

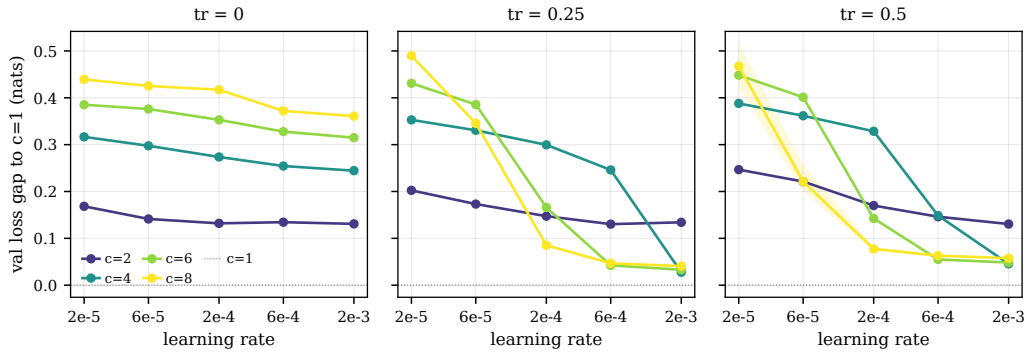


Figure 5: **The paired-data threshold depends on learning rate as well as c and tr.** Validation-loss gap to the c=1 baseline trained at the same learning rate, at 200k steps, for our 14.7M model; our main experiments use 2×10^{-5}. Without paired data, a larger learning rate reduces the gap only slightly. With some (middle, right), the threshold decreases to lower c. c=2, notably, does not benefit.

An initialization-copied model matches baseline sample efficiency, exhibiting no additional validation gap. We read this as an existence proof: a parameter setting where compartmentalization is low-cost exists and is reachable by SGD, but often not from a generic initialization. Refer to Figure 3a for the slowdown comparison.

4.1.2 Post-hoc parameter duplication

We additionally find that it is possible to trivially convert a fully trained c=1 model, after 1M steps, into a $c' \in \{2, 4, 8\}$ model, by duplicating the learned word embeddings and corresponding LM head entries c'-ways. For all c', only 20k additional steps—2% of the original 1M—are required to overcome an initial loss spike, after which the model fully recovers (Fig. 3b).

4.2 Capacity sharing behavior

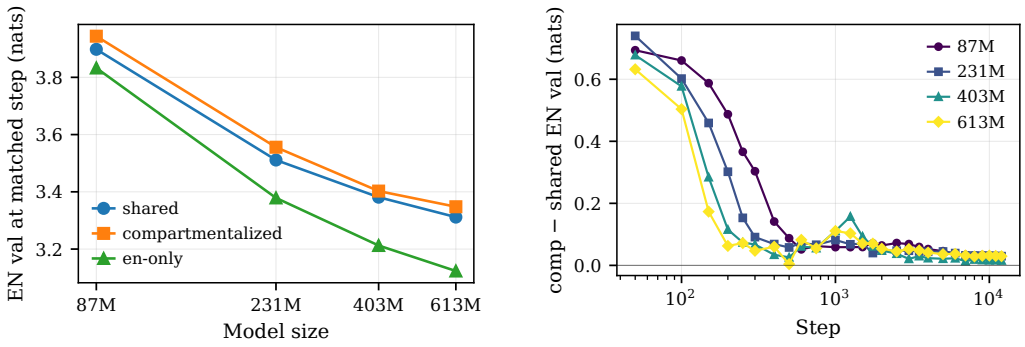
In Figure 2, we evaluate the effect on English validation loss from training simultaneously on an equal number of tokens from other datasets. We find that validation loss suffers almost as badly when the other dataset is English as it does when it is Russian. However, when the other data are text sampled from n-gram estimates of FineWeb, validation loss improves as n decreases. In other words, the simpler the additional data is, the less capacity the model seems to allocate for it. This provides further evidence that the sample efficiency gaps we observe are, in fact, the result of tasks being forced to share a finite amount of model capacity.

4.3 Paired data is often insufficient

We have shown that there are parameter settings that share statistical strength across compartments, but that typical training often fails to find them. We next ask whether paired data reduce compartmentalization.

Taking inspiration from multilingual learning, we encode an experimentally controlled ratio of the dataset as "paired data" (the *paired data ratio*, tr), where the same text is presented twice, with each half encoded in one of two randomly selected compartments, and task tokens preceding each. Our paired data is roughly analogous to parallel corpora in machine translation: each paired example presents the same content in two forms.

Additionally, to avoid requiring the model to spend additional capacity learning c-way compartment identity for common tokens in each compartment, we introduce compartment embeddings, i.e., a d_{model} embedding vector for each of c compartments that we add to the word embedding for each token encoded within that compartment. This includes paired examples, where we add distinct compartment embeddings to each half, including their preceding task tokens. In particular, this removes any ambiguity from the second half of each example, making it completely predictable and



(a) Matched-step EN val loss.

(b) Compartmentalized vs. shared val loss.

Figure 6: **At <1B scale, the gap between multilingual and compartmentalized-multilingual validation loss is small, but there is a consistent sample efficiency gap between the English-only and bilingual models.** a EN validation loss at step 5000 (the last en-only checkpoint) for shared, compartmentalized, and en-only models, across four scales (87M to 613M parameters). b The validation loss gap between training with a single multilingual tokenizer vs. two completely disjoint tokenizers (compartmentalized). The gap between the two closes quickly with steps, suggesting that at least early in training, and for small models, there is high multilingual compartmentalization.

allowing us to use loss convergence toward zero on held-out examples to determine whether the pair-completion task is learned. We also measure *pair-completion accuracy*: given the first half of a held-out paired example, we use the model to greedy-decode the second half, which we score as an exact match.

Results See Figure 4. For our 14.7M parameter configuration, we sweep c and tr, but hold weight decay at 0—we investigate the effect of weight decay in the next section. Surprisingly, we observe that for most c, paired data have almost no effect on decreasing final validation loss, i.e., compartments compete for capacity despite the model quickly learning the pair-completion task (Fig. 4b). Even more surprisingly, we observe a c-dependent phase transition—at our learning rate, at c=8—after which paired data begins to help close the gap.

Learning rate Our main experiments use a small learning rate (2×10^{-5}) for stability during extended overtraining. To test whether the threshold depends on optimization, we repeat a subset of paired-data experiments at learning rates up to 2×10^{-3} (Fig. 5). Without paired data, a larger learning rate does little: at $100\times$ the learning rate the gap shrinks by 15–25%. With more paired data, learning rate helps more, but at c=2, the gap does not improve beyond baseline, even when half of the data is paired examples.

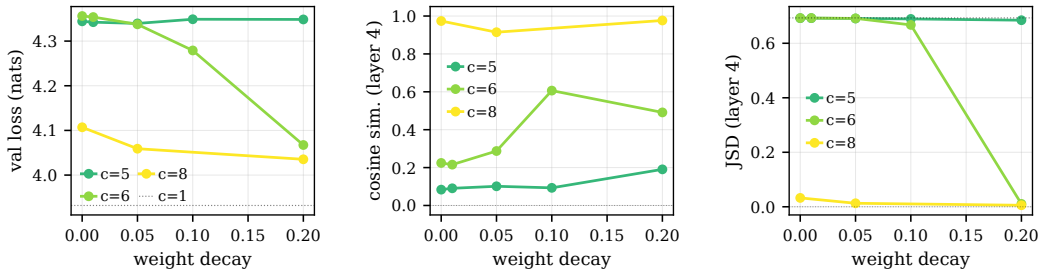
We investigate the effects of weight decay on our 14.7M parameter model next.

4.3.1 Weight decay accelerates the paired data phase transition

In Figure 7, we report the results of additional experiments at the 14.7M scale in which we swept the weight decay hyperparameter $\lambda \in \{0, 0.01, 0.05, 0.1, 0.2\}$. The most striking result is an acceleration of the phase transition, which happens earlier, at c=6, tr=0.75, as λ increases. Additionally, the existing c=8 transition is reinforced and accelerated, with c=8, tr=0.25 at 4.035 nats, lower than any loss achieved at c=2. Outside this interaction with paired data, however, weight decay has little effect at our learning rate: repeating the same sweep without paired data, we find that final validation loss changes by at most 0.03 nats across $c \in \{2, \ldots, 8\}$.

4.4 Measuring bilingual compartmentalization

Pretraining decoders with the architecture of §A.1 at four sizes (87M–613M parameters; §C.2) using the multilingual Qwen3 (Yang et al., 2025) tokenizer on 767M tokens each of English and Chinese Wikipedia text, we find preliminary evidence (Figure 6) that language models compartmentalize



(a) Final val loss. (b) Cross-comp. similarity. (c) Cross-compartment patch JSD.

Figure 7: **For our 14.7M model, at paired data ratio 0.75, weight decay shifts the validation-gap phase transition from c=8 down to c=6.** a Weight decay begins to affect validation loss at a c=6 transition. b Mean per-token cosine similarity between activations at layer 4 of the same text encoded in different compartments. Improvement in representational similarity follows the validation loss trend. c JSD after cross-compartment activation patch at layer 4, the same procedure as shown in Fig. 4d.

languages at smaller scales and suffer a capacity tax vs. single-language training—notice in particular that the shared curve is much closer to the tokenizer-compartmentalized curve, and that this gap decreases with steps, suggestive, given our sample efficiency observations, of low information sharing across surface forms. Further experiments at larger scale are needed to determine whether this trend continues; we did not perform these due to compute constraints.

Findings from Ifergan et al. (2025) and Goldman et al. (2025) provide some evidence that models trained on multiple languages could be compartmentalized, at ∼7B and frontier scales, respectively, but these experiments are insufficient in scope to confirm this hypothesis. The sample efficiency gap between shared and compartmentalized models could widen with scale, and correspondingly the gap between English and multilingual training could close. We present these findings as a validation of the basic characterization of compartmentalization: a training regime in which the model does not recognize consistencies across surface forms.

5 Conclusion

In our construction, language models trained on the same text in c disjoint surface forms learn each form largely independently, exhibiting sample efficiency and capacity costs that increase with c, even though unified solutions exist and even when the model has learned the mapping between forms almost perfectly.

In our experiments, models did not always learn a unified solution, and whether they did depended on learning rate, weight decay, the paired data ratio, and, near the threshold, on the random seed. For pretraining on multiple languages or formats, these results suggest that how well a model learns the mapping from paired data may be an unreliable indicator of how well it is able to transfer across forms.

We offer this as evidence that language models may be inconsistent in their ability to unify internal representations across disjoint surface forms, and propose compartmentalization as a valuable diagnostic lens on a class of representational failures the field has documented at scale.

6 Limitations

We train small models at short context lengths (64 tokens). Further experimentation is needed to determine to what extent our findings generalize to much longer context lengths, like those used in frontier language models.

Additionally, it is possible that compartmentalization is resolved by other aspects of scale, despite the evidence we cite. Further, we accept the simplification of exact token correspondence between

surface forms in order to achieve a controlled experimental setting. Additional experiments are needed to verify whether findings in this regime generalize beyond it. Our main experiments share one conservative learning rate across all scales; we tune it only at 14.7M (§4.3). Several results—the model-scale sweep, the bilingual experiments, and the weight-decay sweep—are single-seed. We train our bilingual experiments on 1.57B tokens, well short of convergence.

See Appendix G for extended discussion on the limitations of compartmentalization.

AI USE STATEMENT

We used AI agents to implement experimental methods and prepare plots, as well as write training and evaluation code. We managed experiments via agents as well. We did not use AI to choose research questions or create the compartmentalization construction. We drafted and structured the paper ourselves, writing the majority of it manually, except for portions of the appendix. AI agents proposed mechanical edits, which we left highlighted until we approved them, typically after extensive manual revision. We have reviewed all results and take full responsibility for the claims made in this paper.

REFERENCES

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GPKTIktA0k.

Nitay Calderon, Eyal Ben-David, Zorik Gekhman, Eran Ofek, and Gal Yona. Empty shelves or lost keys? Recall is the bottleneck for parametric factuality, 2026. URL https://arxiv.org/abs/2602.14080.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, 2020. URL https://aclanthology.org/2020.acl-main.747/.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5f1eee2509599faeeb3570a887016a64-Abstract-Conference.html.

Clément Dumas, Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Separating tongue from thought: Activation patching reveals language-agnostic concept representations in transformers. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31822–31841, 2025. URL https://aclanthology.org/2025.acl-long.1536/.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. Language models scale reliably with over-training and on downstream tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iZeQBqJamf.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9574–9586. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f5c422f4d49a5a807eda27434231040-Paper.pdf.

Omer Goldman, Uri Shaham, Dan Malkin, Sivan Eiger, Avinatan Hassidim, Yossi Matias, Joshua Maynez, Adi Mayrav Gilady, Jason Riesa, Shruti Rijhwani, Laura Rimell, Idan Szpektor, Reut

Tsarfaty, and Matan Eyal. ECLeKTic: A novel challenge set for evaluation of cross-lingual knowledge transfer, 2025. URL https://arxiv.org/abs/2502.21228.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Jing Huang, Daniel Wurgaft, Rachit Bansal, Laura Ruis, Naomi Saphra, David Alvarez-Melis, Andrew Kyle Lampinen, Christopher Potts, and Ekdeep Singh Lubana. Why larger models learn more: Effects of capacity, interference, and rare-task retention, 2026. URL https://arxiv.org/abs/2605.29548.

Maxim Ifergan, Leshem Choshen, Roee Aharoni, Idan Szpektor, and Omri Abend. Beneath the surface of consistency: Exploring cross-lingual knowledge representation sharing in LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4630–4644, 2025. URL https://aclanthology.org/2025.findings-naacl.475/.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017. doi: 10.1162/tacl_a_00065. URL https://aclanthology.org/Q17-1024/.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. Cross-lingual ability of multilingual BERT: An empirical study. In *The Eighth International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJeT3yrtDr.

Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022.

Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Jirui Qi, Raquel Fernández, and Arianna Bisazza. Cross-lingual consistency of factual knowledge in multilingual language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10650–10666, 2023. URL https://aclanthology.org/2023.emnlp-main.658/.

Anton Schäfer, Shauli Ravfogel, Thomas Hofmann, Tiago Pimentel, and Imanol Schlag. The role of language imbalance in cross-lingual generalisation: Insights from cloned language experiments, 2024. URL https://arxiv.org/abs/2404.07982.

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

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do Llamas work in English? On the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15366–15394, 2024. URL https://aclanthology.org/2024.acl-long.820/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

A Architecture and training setup

A.1 Model architecture

All models are GPT-style decoders following the nanoGPT (Karpathy, 2022) lineage with two modifications: (a) RoPE (Su et al., 2024) replaces learned absolute position embeddings, and (b) embedding and LM-head matrices are not tied. We use pre-LayerNorm, GELU MLPs with $4d_{\text{model}}$ hidden width, causal attention with d_{head}=32 (so $n_{\text{head}} = d_{\text{model}}/32$), block size T=64, and no dropout. Table 1 summarizes parameter counts at c=1 and c=8 across the model-size ladder.

A.2 Optimizer and schedule

We use AdamW with β_1=0.9, β_2=0.999, ε=10^{-8}, and a peak learning rate of 2×10^{-5} reached via 1,000 iterations of linear warmup; learning rate is then held constant for the remainder of training, with no decay. We train for 1,000,000 iterations at the small-scale ladder (8-32 through 8-512), using bfloat16 mixed precision throughout. The effective batch size is held at $2048 \times T = 131{,}072$ tokens per optimizer step across all small-scale runs, with per-step batch size and gradient-accumulation factor varied to fit in memory for each (d_{model}, c) pair on a single 80 GB GPU. Weight decay is 0 except in the sweep of §E.1.

A.3 Tokenizer

For the worst-case-model and intervention experiments we train a BPE tokenizer of vocabulary size V=16,384 on a subsample of FineWeb (Penedo et al., 2024) using SentencePiece's BPE implementation. The compartmentalized composite vocabulary is $cV + 1$, where the $+1$ is a single shared cross-compartment pairing marker that we prepend to paired sequences. The bilingual experiments (§4.4) instead use the Qwen3 tokenizer (Yang et al., 2025) ($V \approx$152k).

Table 1: Model sizes for the compartment baseline experiments, broken down by width and compartment count c. The composite vocab in the c-compartment setting is $cV + 1$ (the $+1$ is a task token). "Trunk" is all non-embedding parameters (attention, MLP, and LayerNorm gains); "Embedding" is the word-embedding matrix plus the untied LM head, both of which are sized $cV + 1$. Total is exactly Trunk + Embedding. **The trunk is the same for every c; compartmentalization adds only compartment-specific embedding and LM-head rows.** The embedding share of parameters therefore increases with c and decreases with width. Intermediate $c \in \{3, 5, 6\}$ are listed at 8-256, the scale at which the full compartment sweep is run.

Config	L	d_{model}	c	Total	Trunk	Embedding	Emb. %
8-32	8	32	1	1.1 M	0.1 M	1.0 M	91.4%
			2	2.2 M	0.1 M	2.1 M	95.5%
			4	4.3 M	0.1 M	4.2 M	97.7%
			8	8.5 M	0.1 M	8.4 M	98.8%
8-64	8	64	1	2.5 M	0.4 M	2.1 M	84.2%
			2	4.6 M	0.4 M	4.2 M	91.4%
			4	8.8 M	0.4 M	8.4 M	95.5%
			8	17.2 M	0.4 M	16.8 M	97.7%
8-128	8	128	1	5.8 M	1.6 M	4.2 M	72.7%
			2	10.0 M	1.6 M	8.4 M	84.2%
			4	18.4 M	1.6 M	16.8 M	91.4%
			8	35.1 M	1.6 M	33.6 M	95.5%
8-256	8	256	1	14.7 M	6.3 M	8.4 M	57.1%
			2	23.1 M	6.3 M	16.8 M	72.7%
			3	31.5 M	6.3 M	25.2 M	80.0%
			4	39.9 M	6.3 M	33.6 M	84.2%
			5	48.2 M	6.3 M	41.9 M	86.9%
			6	56.6 M	6.3 M	50.3 M	88.9%
			8	73.4 M	6.3 M	67.1 M	91.4%
8-512	8	512	1	42.0 M	25.2 M	16.8 M	40.0%
			2	58.7 M	25.2 M	33.6 M	57.1%
			4	92.3 M	25.2 M	67.1 M	72.7%
			8	159.4 M	25.2 M	134.2 M	84.2%

B Compartmentalization construction

B.1 Vocabulary offset scheme

Given a base BPE tokenizer of vocabulary size V, the c-compartment composite vocabulary is constructed by replicating the base vocabulary c times with offset: token id t encoded under compartment j becomes $jV + t$, where $j \in \{0, \ldots, c-1\}$. Embedding and LM-head rows are initialized independently across compartments, so at initialization nothing in the parameters indicates that the c blocks of the embedding matrix correspond to the same tokens.

B.2 Compartment embeddings

To prevent the model from spending capacity to learn compartment identity from token co-occurrence, we add a learned d_{model}-dimensional bias to the input embedding of every token according to its compartment, in addition to the offset described above. Compartment embeddings are initialized to zero and applied uniformly within each compartment, including paired examples (each half receives the bias of its own compartment). This makes the second half of each paired example fully predictable in principle, allowing us to use convergence of the second-half loss toward zero as a check that the pair-completion task is being learned.

C DATASETS

C.1 FINEWEB PRETRAINING CORPUS

For the worst-case-model and intervention experiments we use a deduplicated subsample of FineWeb (Penedo et al., 2024), tokenized with the BPE-16384 tokenizer described above. Train and validation shards are disjoint document-level. Across all small-scale runs, training consumes approximately $131{,}072 \times 10^6 \approx 131$ billion tokens—between $\sim$160$\times$ and $\sim$5000$\times$ Chinchilla-optimal for the model sizes in Table 1.

C.2 ENGLISH AND CHINESE WIKIPEDIA

For the bilingual experiments (§4.4) we use a 767M-token English subsample and a 767M-token Mandarin Chinese subsample of Wikipedia, both tokenized with the Qwen3 tokenizer. The shared condition uses the single Qwen3 vocabulary. The compartmentalized condition offsets every Chinese token ID by the vocabulary size (151,936), so the two languages share no tokens, as in §B.1. Models use the architecture of §A.1 with untied embeddings, block size 512, and $(n_{\text{layer}}, d_{\text{model}}) \in \{(12, 256), (24, 512), (24, 768), (24, 1024)\}$: 87M, 231M, 403M, and 613M parameters with the shared vocabulary, of which 9.4M, 75.5M, 170M, and 302M are non-embedding; the compartmentalized models have twice the embedding parameters. We train with AdamW at $256 \times 512 = 131{,}072$ tokens per step, peak learning rate 3×10^{-4} after 1,000 warmup steps, and cosine decay to 3×10^{-5} over each run: 12,000 steps (1.57B tokens, about one epoch of both languages) for the bilingual runs and 5,852 steps (one epoch of English) for the English-only runs. All runs use one seed, trained on a single GPU.

D EVALUATION METHODOLOGY

D.1 PER-COMPARTMENT VALIDATION LOSS

At each evaluation, we sample a fixed-seeded validation batch and compute cross-entropy at every token position, restricted to positions assigned to a single compartment. The reported *val loss* for a c-compartment model is the unweighted mean of the per-compartment losses—equivalent to the loss the model would achieve on a $1/c$-fraction validation set drawn from each compartment in turn. For c=1 this reduces to the standard FineWeb validation loss.

D.2 TARGET-HALF LOSS FOR PAIRED EXAMPLES

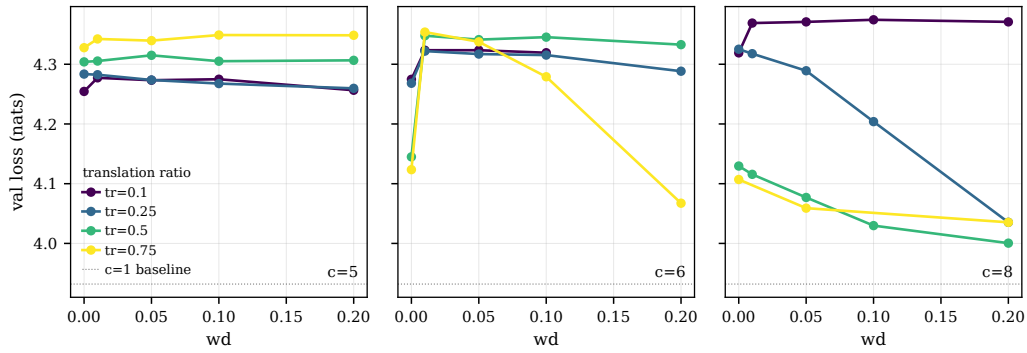
On paired evaluation examples, we report the cross-entropy on the *target* half of each pair only (i.e., predicting compartment-j tokens given a compartment-i source for $i \neq j$). This isolates whether the model has learned the pair-completion task itself, separately from whether paired training data has reduced compartmentalization in the per-compartment val loss. We average over all $i \neq j$ ordered pairs.

D.3 EVALUATING REPRESENTATIONAL SIMILARITY

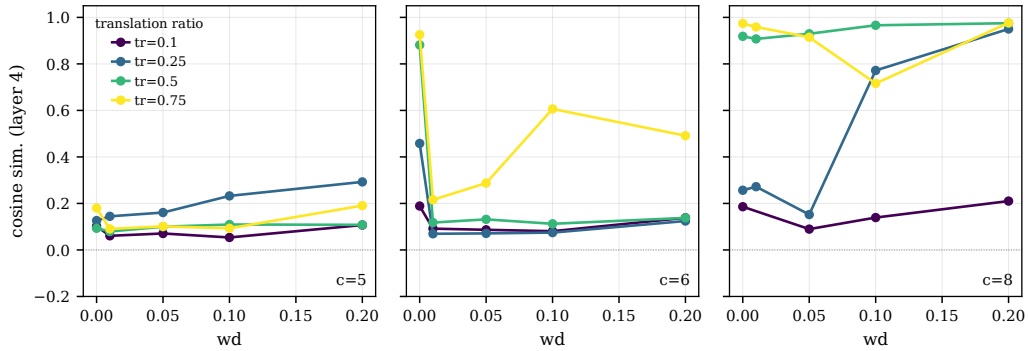
We measure cross-compartment representational alignment using per-token mean cosine similarity. For a fixed batch of B=64 FineWeb val token sequences of length T=64, and for each compartment $j \in \{0, \ldots, c-1\}$, we run the model on $\mathbf{x}_j = \mathbf{x} + jV$ (with compartment-j compartment embeddings) and record the post-block hidden state at layer 4 (counting from 0). For each unordered compartment pair (i, j) we compute the mean per-row cosine similarity between the corresponding $(B \cdot T, d_{\text{model}})$ activation tensors, then average across all $\binom{c}{2}$ pairs to obtain the reported mean cosine similarity.

D.4 VALIDATION BATCH AVERAGING

All reported validation losses are means over N_{eval}=100 independent $B \times T$ batches per checkpoint, sampled from the held-out validation shards.



(a) Validation loss.



(b) Cross-compartment cosine similarity.

Figure 8: **Full weight-decay sweep at the 14.7M scale, for $c \in \{5, 6, 8\}$ across tr $\in \{0, 0.056, 0.167, 0.25, 0.5, 0.75\}$ and wd $\in \{0, 0.01, 0.05, 0.1, 0.2\}$.** Body Fig. 7 shows the tr=0.75 slice. Weight decay has no visible effect at c=5 for any paired data ratio; at c=6 it lowers loss only at tr=0.75 for wd$\geq$0.1; at c=8 it lowers loss further for tr$\geq$0.5.

For the bilingual experiments (§C.2), per-language validation losses use B=4 sequences of T=512 tokens averaged over N=200 contiguous chunks of the held-out per-language val shards, for 409,600 prediction tokens per language per checkpoint.

E Full sweep results

E.1 Weight decay × paired data ratio × c

See Fig. 8.

E.2 Interaction between learning rate and performance gap

See Fig. 9.

F Downstream task performance

Models train at a 64-token context, and longer evaluation sequences are left-truncated. We therefore report the three tasks whose scored sequences mostly fit: 99.9% for SciQ, 92.3% for ARC-e, and 86.1% for PIQA. We exclude HellaSwag (30.6%) and LAMBADA (2.9%). Every model is truncated identically, so comparisons across c and paired data ratio are unaffected.

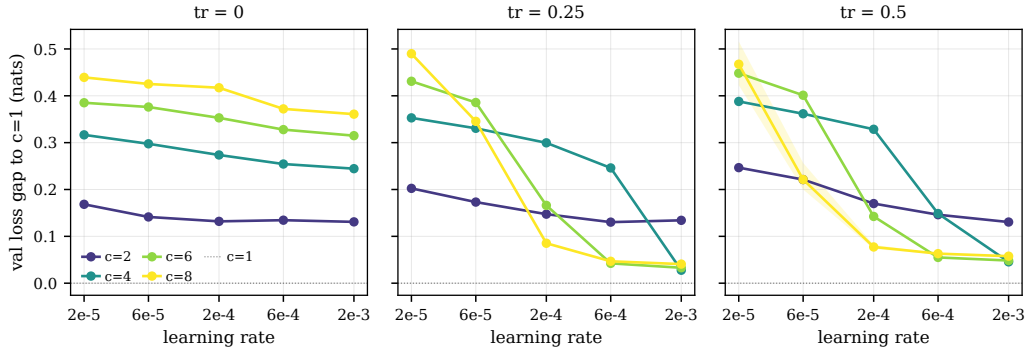


Figure 9: **The paired-data transition point depends on learning rate, as well as c and tr.** Validation loss gap to the c=1 baseline trained at the same learning rate, at 200k steps. Our main experiments use 2×10^{-5}. The shaded region spans the min. and max. of 3 seeds for c=8 at tr=0.5 to show seed dependence.

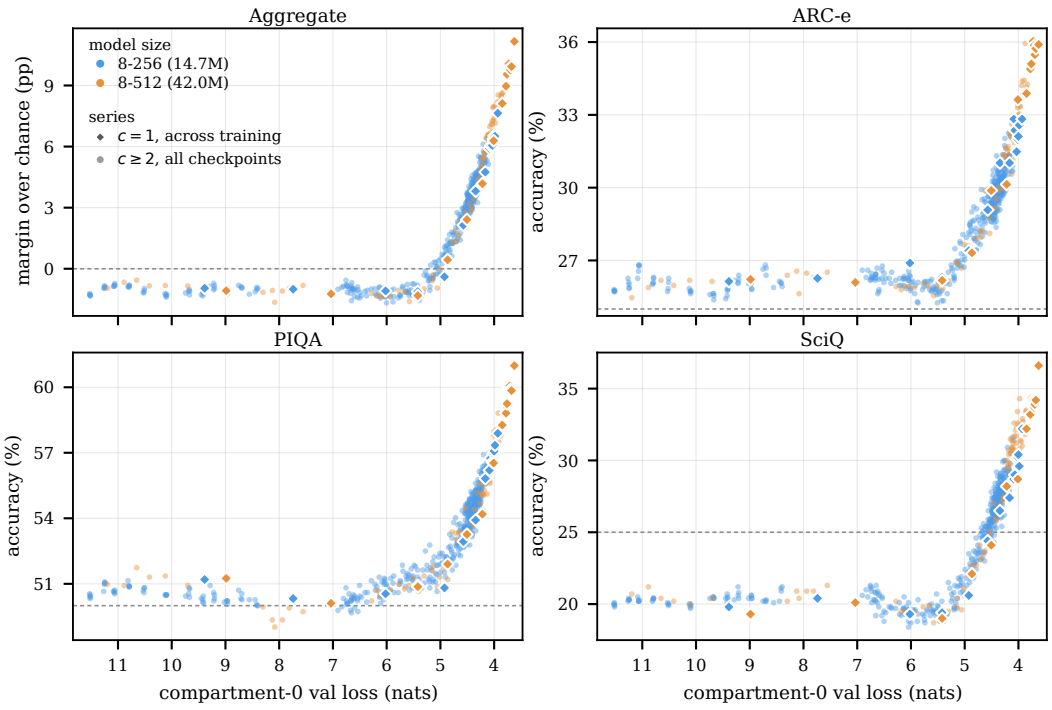
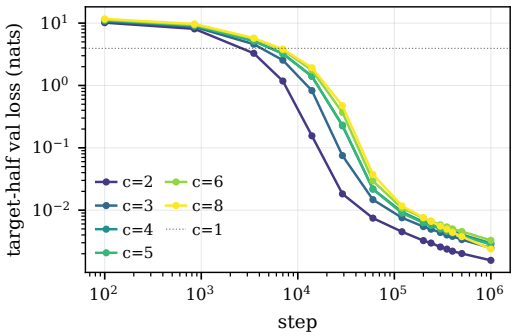


Figure 10: **Downstream accuracy is a function of validation loss, even under compartmentalization.** Each dot is one checkpoint of a $c \geq 2$ run, across every compartment count and paired data ratio at that size; each diamond is a checkpoint of that size's c=1 baseline. The dotted line marks random choice. Scoring picks the highest-likelihood option rather than guessing uniformly, so a model that has not yet learned the task is systematically rather than randomly wrong, and can score below chance, as on SciQ.

Following the loss-to-performance protocol of Du et al. (2024) and Gadre et al. (2025), we plot downstream task accuracy directly vs. validation loss (Fig. 10). For each model size, a compartmentalized model scores about as well as any other model that reaches the same loss, including the c=1 baseline. Compartmentalization therefore affects downstream accuracy through validation loss; we find no additional downstream effect of our construction.



(a) 14.7M–73.4M, tr=0.5.

Figure 11: **Target-half validation loss is well below the c=1 validation loss at every compartment count.** a One line per c at fixed tr=0.5, weight decay 0; the dotted line is the c=1 validation loss. Every line falls below 0.04 nats by step ∼60k and continues toward ∼0.002 nats.

G Extended Limitations

Our notion of compartmentalization is quite simplified, and does not allow for multiple overlapping types of surface forms, i.e., under this scoping, surface form must be both discrete and disjoint. We make no claim as to how much compartmentalization as we characterize it contributes to the overall difficulty of modeling language. In addition, while we performed experiments to validate our characterizations of sample efficiency and capacity in our controlled setting, we do not claim that similar behavior in other settings is conclusively or likely caused by representational redundancy.

H Target-half loss on paired examples

Plotted in Fig. 11 is validation loss for the deterministic second half of held-out paired examples; as such, we expect it to approach zero. This loss continues to decrease after pair-completion accuracy saturates (Fig. 4).

I n-gram second compartments

The n-gram conditions in Figure 2 fill compartment 1 with text sampled from an n-gram model estimated on the same FineWeb corpus the models train on, so that the second compartment's data becomes progressively more English-like as n rises. This section describes how those models were estimated and sampled.

We estimate models for $n \in \{2, 3, 4\}$ from 1B tokens using the same V=16,384 tokenizer the models train on, and sample compartment 1 ancestrally from them. Contexts absent from the table back off to order $n-1$: held-out coverage is 100%, 96% and 82% at n=2, 3, 4, and backoff at n=4 affects about 0.2% of tokens. The unigram-frequency compartment is the n=1 member of the same family.

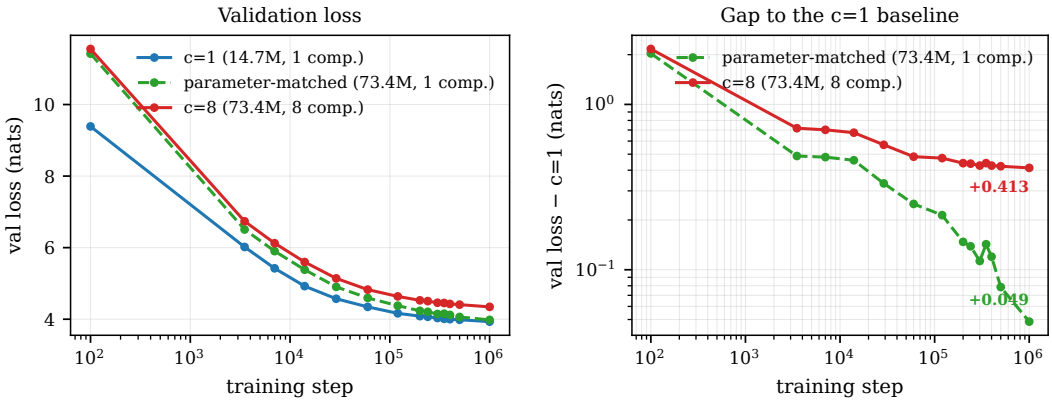
J Existence proofs and post-hoc duplication

J.1 Initialization-copy

In Fig. 3, we show the training trajectories of embedding copy experiments. In Fig. 13, we validate that embedding copying works across scales.

J.2 Post-hoc parameter duplication trajectories

See Fig. 14.



(a) Validation loss.

(b) Gap to the c=1 baseline.

Figure 12: **Parameter-matched control.** A model with the same enlarged embedding as c=8, but trained on one compartment's data. At 10^6 steps its gap to the c=1 baseline is $+0.0485$ nats, compared with $+0.4132$ for c=8.

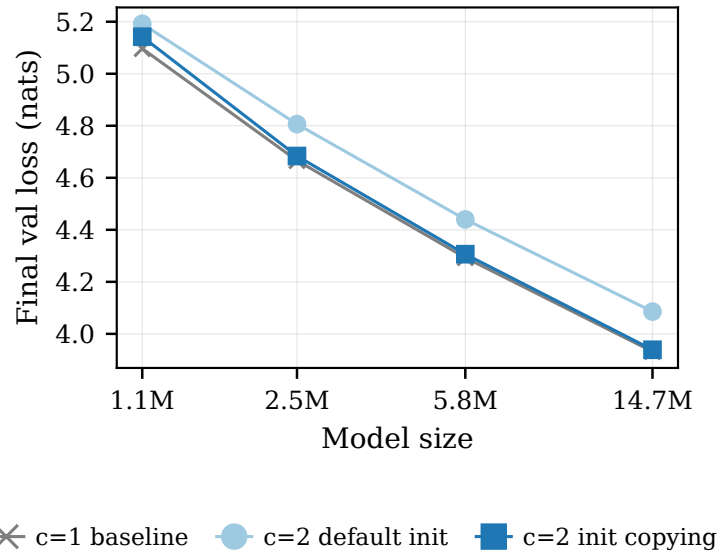


Figure 13: **Initialization-copy results across** $d_{\text{model}} \in \{32, 64, 128, 256\}$**.** At every scale, c=2 with init-copy closely follows the c=1 baseline; default-initialized c=2 shows a persistent compartmentalization gap. Body Fig. 3 shows the 14.7M instance only.

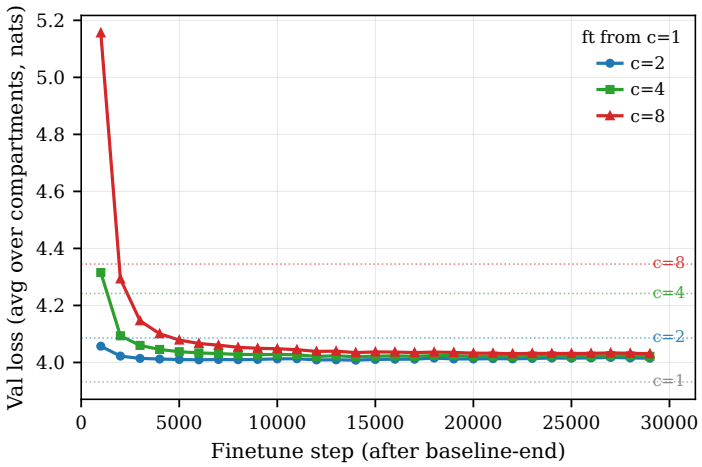


Figure 14: **Post-hoc parameter duplication, full per-c trajectories at the 14.7M scale.** A c=1 checkpoint at step 10^6 is converted into a $c \in \{2, 4, 8\}$ model by duplicating learned word embeddings and LM-head entries. Each curve shows compartment-averaged val loss over the subsequent 20,000-step finetune; loss returns to its pre-duplication level within ∼2,000 steps and every c matches the c=1 baseline thereafter.