Is your LLM a sequence model on the training history? The origins and consequences of anticipation

Szilvia Ujváry* SRU23@CAM.AC.UK
University of Cambridge

Bruno Kacper Mlodozeniec BKM28@CAM.AC.UK
University of Cambridge

José Miguel Hernández-Lobato JMH233@CAM.AC.UK
University of Cambridge

Ferenc Huszár FH277@CAM.AC.UK
University of Cambridge

Dmitrii Krasheninnikov* DMKR0001@GMAIL.COM

Abstract

Recent work showed that, in cyclic fine-tuning settings, transformers exhibit striking *anticipatory recovery*: the loss on an upcoming document decreases *before* that document is revisited. We extend this phenomenon beyond individual-document cycling to bursty pre-training over mixtures of class-conditional distributions, where examples from the same class appear in predictable bursts – so the model is trained on one class for several steps, then on the next one, and so on. We show that Transformers anticipate upcoming classes despite having no explicit memory of training history, and that *anticipation is linked to a lower training loss*. In our experiments, models realize a substantial fraction of the anticipatory advantage achievable by sequence models that explicitly learn class order. Additionally, we demonstrate anticipation in a setting where the next class can only be predicted by knowing the previous two classes, suggesting that Transformers can partially behave like sequence models over the training stream itself. To study the optimization dynamics underlying anticipation, we analyze the implicit bias of bursty mini-batch training and identify an implicit bias aligning the gradients of temporally adjacent classes. Finally, we show that structured training order reshapes measured relationships between classes, suggesting that training order itself can act as a source of representational bias during optimization.

1. Introduction

Modern language models are trained under the assumption that the order in which training documents are presented (the training order) does not matter. Yet recent work suggests that Transformers [30] can retain information about the training order in their activations despite having no explicit memory of past updates [14]. Yang et al. [33] further showed that, in cyclic fine-tuning, where documents are revisited in deterministic cycles, models exhibit striking anticipatory recovery: the loss on a document decreases *before* that document is revisited, indicating that the model anticipates it.

The phenomenon of anticipation is surprising because it is consistent with behavior we expect of *sequence models* over the training stream. In this work, we study how anticipation extends beyond repeating individual documents, using a structured setup where only the upcoming domain can be predictable from the training order. Concretely, we arrange mixture data in class-homogeneous batches, and train on each class for several steps before moving to another class, hereby creating class bursts. Similar forms of predictable temporal structure naturally arise in continual learning [5, 12, 17],

Equal contribution. © S. Ujváry, B.K. Mlodozeniec, J.M. Hernández-Lobato, F. Huszár & D. Krasheninnikov*.

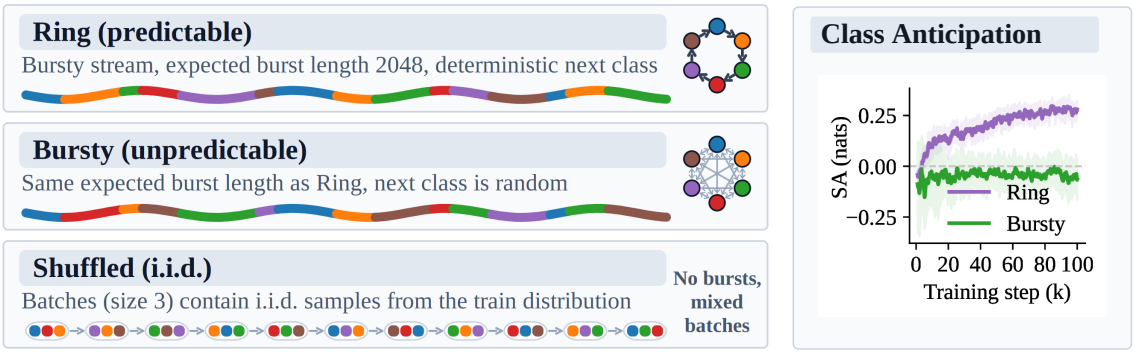

Figure 1: **Surprising class anticipation during training.** *(Left) Experimental setup.* In Ring and Bursty settings, the model is trained in bursts – multiple updates in a row on data from the same single class (32 steps on average) – meaning the training history is not i.i.d. Transition structure between bursts is defined by a Markov chain (see trainsition diagrams on the right). Colors encode classes; wavy strips show example training streams. We show 6 classes for readability; experiments use 10. *(Right) Anticipation measured with Successor Advantage (SA) over time*. Roughly, SA measures how much the eval loss on the actual successor class is lower than that on the other non-current classes. Ring SA keeps increasing, meaning the **model learns to anticipate which class is coming next** before being trained on that class, despite no explicit memory of the training stream.

curriculum learning [3], and reinforcement learning [28]. We additionally identify an implicit bias of bursty mini-batch training that aligns temporally adjacent class gradients. Our contributions are:

1. We demonstrate anticipation in bursty pre-training streams, extending prior work on anticipation of individual documents to anticipation of document class distributions. We also show that Transformers anticipate even under second-order transition structure (Sections 2 and 4).
2. We show that predictable training order provides a measurable advantage on training loss. Cyclic training realizes a substantial fraction of the gain achievable by oracle class-conditioning, suggesting partial sequence-model-like behavior over the training stream itself (Section 2).
3. We provide a theoretical analysis of bursty mini-batch training and identify an implicit bias aligning adjacent-class gradients. Our result provides predictions qualitatively consistent with observed optimizer and scaling dependencies of anticipation (Sections 3 and 4).
4. We show that ordered training reshapes relationships between class distributions, suggesting that training order can act as a source of representational bias during optimization (Section 4).

2. The anticipation phenomenon

Prior work studied anticipatory recovery in cyclic document fine-tuning [33]. We instead study bursty pre-training streams, where class distributions repeat and recur in a structured way. We use *anticipation* to refer to the general phenomenon, *anticipatory recovery* [33] for the metric used in prior work, and *Successor Advantage* (SA) as our primary metric.

Training order versions. We focus on cyclic orderings. Our main setting is *Ring*, where data is organized in class-homogeneous bursts (sequence of batches) of geometrically distributed length, and after the burst ends, we deterministically transition to a successor class. We compare to *Bursty (unpredictable)* and *shuffled* baselines: the former has class-homogeneous bursts of the same as Ring,

but the order of classes is unpredictable, the latter is fully i.i.d. with mixed-class batches. Figure 1 (left) illustrates our setup. More details are in Section B.4.

While anticipatory recovery measures the loss trajectory of a repeated document, we seek a quantity that emphasizes adaptation to the structure of the training stream itself. We therefore define *Successor Advantage* (SA), which measures whether the model assigns preferentially low loss to the upcoming class relative to other non-current classes.

Definition 2.1 (Successor Advantage (SA)) *Let $\mathcal{D}_{eval}$ be a held-out dataset drawn i.i.d. from $p(x) = \sum_{c \in C} p(x \mid c)p(c)$. At step t of training, let us denote by $c_{current}$ the set of elements of the class within $\mathcal{D}_{eval}$, whose batch the model has most recently been trained on, and by c_{next} the set of elements of the upcoming class after the current one. The successor advantage at step t is:*

$$SA_t = \frac{1}{|\{x : x \notin c_{current} \cup c_{next}\}|} \sum_{x \notin c_{current} \cup c_{next}} \ell(x) - \frac{1}{|c_{next}|} \sum_{x \in c_{next}} \ell(x)$$

Our experiments also use a corrected version of the SA to remove biases stemming from differing class difficulty (see Section B). Positive SA indicates that the model is preferentially adapted to the future training distribution, rather than just tracking the class currently being optimized. However, positive SA does not by itself explain why optimization should produce anticipation. It could simply arise as an incidental consequence of cyclic training or representational overlap. The following proposition shows that, under bursty training, adaptation to the future training distribution can provide a direct optimization advantage.

Proposition 2.2 *Let C denote the set of classes and M the sequence length. The best possible average loss improvement over shuffled training via anticipation is $\log |C|/M$. This is achieved by sequence models explicitly modeling class order, i.e. $p_{sequence\ model}(x) = p(x \mid c : x \in c)$.*

This result reframes anticipation as more than a curious training artifact: *model are incentivized to anticipate, because this allows them to decrease the training loss further*. The proposition also gives us a way to test to what extent do models behave like sequence models.

Experiments. Our main experiments use a ∼20M parameter decoder-only Transformer [30] model trained with AdamW and the standard next-token prediction objective for 100k steps on 10-language Wikipedia data. Batch size is 64, and there are on average 32 gradient steps per class burst (so the mean burst length is 2048 samples). Unless stated otherwise, the context length is 8 tokens (excluding BOS and EOS) and class orderings vary with the random seed. See Section B for details. Figure 1 (Right) shows increasing SA in Ring training, whereas Bursty (unpredictable) does not anticipate. Figure 5 in Section A shows that high SA correlates with increased anticipatory recovery.

Comparison to sequence models. Figure 2 shows that Ring training realizes about 30% of the possible anticipatory loss advantage (the difference between the Shuffled and Theoretical limit), achievable by sequence models explicitly learning class order. To test whether the rest of possible anticipatory loss advatage is absent due to limitations of our empirical setup (such as model size, training horizon, etc.), we estimated an *empirical limit* of anticipation, computed as follows: during

training, models received the true class index prepended to each training sample, artificially "conditioning" the model on the class in-context. These class cue tokens were excluded from training loss computation. The experimental setup was otherwise identical. The training loss of this oracle model was taken as an empirical estimate of the best possible anticipatory loss advantage achievable with the specific model architecture and training horizon we considered. Our transformer realized 40% of this advantage. Details are in Section D.1. We note that anticipatory advantage on the training loss comes at the expense of larger validation loss. This is not surprising, as models that anticipate adapted themselves heavily to the training history, while the validation distribution is a mixture, which creates a distribution shift for these models.

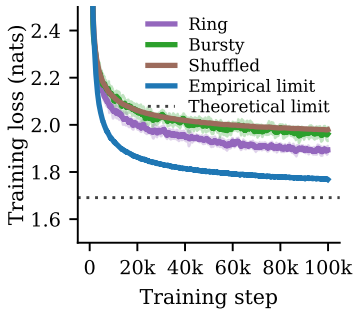

Figure 2: **Anticipation is linked to lower training loss.** Lower than "Shuffled" indicates sequence model-like behavior.

3. Anticipation as the implicit bias of bursty mini-batch training

Towards explaining anticipation, Yang et al. [33] proposed a toy model of training which reproduces anticipatory recovery. We instead give a related, but distinct explanation: anticipation can arise from implicit biases induced by bursty mini-batch training. To make this intuition analytically tractable, we derive the implicit bias of SGD in bursty settings, showing that cyclic mini-batch updates create a consistent pressure toward alignment of temporally adjacent class gradients. A related gradient alignment mechanism was hypothesized informally by Krasheninnikov et al. [13] and Nichol et al. [20]; we prove a version for class-conditional batches. Our derivation builds on Roberts [23], who studied the implicit bias of SGD in shuffled training. We explain our result informally, and refer the reader to Section C.2 for full definitions and proofs.

Setup (informal). We compare two sequential SGD updates on bursty mini-batches B_{current} and B_{next} against the equivalent full-batch GD updates on $B_{\text{current}} \cup B_{\text{next}}$. In bursty training, each mini-batch contains samples from only a single class, denoted c_{current} and c_{next}. Let $G^{(\text{current})}$, $H^{(\text{current})}$, and Σ^{current} denote the expected class gradients, Hessians, and gradient covariance matrix, respectively. Denoting by $\delta g_i^{\text{bursty}}$ the deviation of SGD from the corresponding GD trajectory in parameter i, the following theorem characterizes the expected drift induced by bursty mini-batch updates.

Theorem 3.1 (Implicit bias of bursty training) *Let $B_{current} \overset{i.i.d.}{\sim} p_{current}(x)$ and $B_{next} \overset{i.i.d.}{\sim} p_{next}(x)$ be two minibatches of size $N/2$. Assume that $\operatorname{tr}\Sigma^{(current)} \approx \operatorname{tr}\Sigma^{(next)} = \operatorname{tr}\Sigma_{within}$. Then, to leading order in η, with expectation is taken over the randomness in the minibatches $B_{current}$ and B_{next} :*

$$\mathbb{E}[\delta g_i^{bursty}] = -\frac{\eta}{2N}\,\partial_i \operatorname{tr}\Sigma_{within} - \frac{\eta}{8}\,\partial_i \|G^{(current)} - G^{(next)}\|^2 - \frac{\eta}{2}\left(G_j^{(next)} H_{ij}^{(current)} - G_j^{(current)} H_{ij}^{(next)}\right).$$

Interpretation. Theorem 3.1 shows that bursty SGD is equivalent, to leading order, to GD on an objective augmented by two additional implicit regularization terms (Term 1 $\propto \partial_i \operatorname{tr}\Sigma_{\text{within}}$ and Term 2 $\propto \partial_i \|G^{(\text{current})} - G^{(\text{next})}\|^2$) and a generally nonconservative curvature interaction term (Term 3). Crucially, the main anticipation-driving term is Term 2, which explicitly favors optimization trajectories where adjacent classes acquire increasingly aligned gradients. In cyclic bursty training, this alignment term is repeatedly applied to adjacent class pairs, and thus can make the current class become progressively more beneficial for its successor, producing anticipation. The remaining terms

appear to modulate this effect. Term 1 suppresses within-class gradient variance and weakens with increasing batch size, but does not directly align consecutive classes. While Term 3's optimization role is less clear, its dependence on curvature suggests its role may depend on the optimizer used.

Although the derivation applies specifically to SGD, the resulting decomposition provides a way to compare how different optimizers modulate the above drift terms. Measuring their relative contributions yields concrete predictions about which optimizers should exhibit anticipation, which we test empirically in the next section.

4. How far does anticipation extend?

The implicit bias interpretation identifies a mechanism by which anticipation can emerge under cyclic bursty training – but how general and sequence-model-like is this behavior in practice? Here we study anticipation across transition structures, optimizers, scales, and training configurations. Section D contains further results including image modeling (Section D.4) and PCA analysis (Section D.2).

Second-order transitions (Section D.3). We showcase smaller-scale, but present anticipation in second-order settings, where the next class is inferrable only from the pair (previous class, current class).

The transition table is a Latin square. We test a 5-class (Figure 3) and 6-class version (Section D.3),which have 25 and 36 states, respectively. For any fixed current class, each possible next class appears exactly once across predecessor classes. This makes the first-order marginal uninformative: knowing only the current class gives a uniform distribution over next classes. A model can do better than chance only if training has left it with usable information about the previous class too. Averaged across 16 seeds, corrected SA is positive: $+0.0412 \pm 0.0081$, demonstrating small but significant anticipation.

The above experiment demonstrates that transformers can partially behave like sequence models over the training stream even beyond simple first-order transition structure.

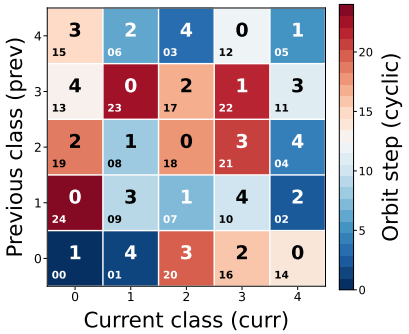

Figure 3: **Latin-squared 2nd order transition function.** x and y axis labels show the individual state index. Colors and small-sized numbers in each cell show the index of state pairs (previous, current). Large-sized digits denote the next class from that cell pair. The orbit starts from the bottom left.

Optimizer (Section D.5, Table 5). We tested SGD, AdamW, AdagradW, Lion and Muon. Only AdamW and AdagradW showed anticipation (positive SA), despite Lion and Muon also training well. Our experiments in Section D.5 suggest that AdamW encourages anticipation through suppressing Term 3 in Theorem 3.1 via second moment preconditioning. Further, we find that anticipation generally increases with the second moment β_2 and happens even without momentum (β_1), suggesting that β_2 is the main driver of the phenomenon. This coincides with intuition as larger β_2 causes more historical gradients to influence parameter updates. It is important to note that Yang et al. [33] observed anticipatory recovery for SGD too. An explanation of the discrepancy could be that same-domain (CNN news) individual-documents naturally have more similar Hessians than samples from different languages, and that pretraining and scale further reduces the Hessian mismatch.

Scaling axes. Ablating the **Expected number of gradient steps per burst (Figure 10)**, we observe increasing anticipation up to an expected 32 steps per class, consistent with Yang et al. [33]. This is

in line with Theorem 3.1 as more gradient steps allow class-gradient alignment to exercise its effect more. However, increasing the number of steps further is detrimental, likely because adapting to the current class becomes the advantageous behavior. This is shown by Bursty's training loss approaching Ring's. We find that larger **Batch size (Figure 12),** generally helps anticipation, though the difference between 64 and 256 in SA is negligible. Larger batch size allows for better gradient mean estimates, helping class alignment. In addition, Term 1 in Theorem 3.1 decreases with increasing batch size, hence any possible interference from this term is reduced. Varying the **Context length (Figure 11),** we find that, contrary to Yang et al. [33] who used individual documents instead of classes, longer context lengths hurt anticipation, likely because the class becomes easily identifiable from a prefix, hence any anticipation advantage persists only for a few tokens. The theoretical room for anticipation also decreases. Finally, increasing **Model size (Figure 13)** increases the anticipation effect. While Yang et al. [33] only showed anticipation for model scales starting at 160M parameters, we showcase non-negligible anticipation for as small as 0.19M parameter models.

5. Consequences of anticipatory inductive bias

To demonstrate the extent to which anticipatory inductive bias shapes training, we investigate inter-class relative gradient alignments under fixed Ring orders. We find that a-priori gradient alignments are significantly altered: a striking example is Spanish becoming more aligned to Finnish than the a-priori similar Portuguese. Figure 4 shows excess cosine similarities after 200k training steps in the Ring order displayed (starting with English, ending with Czech). For each current class i and next class k, cell (i, j) reports $Sim(G^{(i)}, G^{(k)}) - \sum_{j \notin \{i,k\}} Sim(G^{(i)}, G^{(j)}))$, i.e. the alignment of class i to k, relative to all other classes.

After training, relative alignments no longer respect a-priori class similarities, but the training order (see red off-diagonals). This demonstrates that training order can intervene on the relationships between classes learned by the model. More details are Appendix Section D.7.

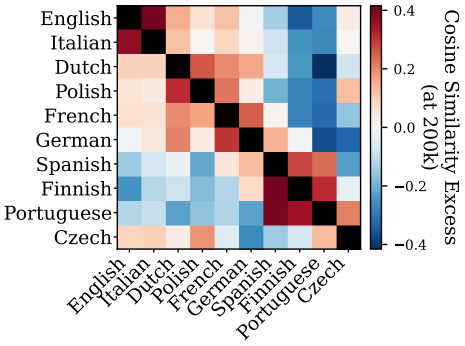

Figure 4: **Relative gradient cosine similarities** at the end of training (200k steps), when using the fixed Ring order matching the order of the figure labels (up to bottom, left to right).

6. Discussion

We showed that Transformers anticipate the upcoming classes in cyclic bursty pre-training, and realize a substantial fraction of the anticipatory advantage achievable by sequence models over the training stream. We provided a possible explanation for this phenomenon by identifying an implicit bias of bursty mini-batch training, that alignins adjacent class gradients.

Our results suggest that training order is not an easy-to-ignore optimization detail: cyclic bursty training promotes alignment between temporally adjacent classes, reshaping relationships between them. As anticipation strengthens with scale, increasingly weak temporal structure may become sufficient to induce such effects beyond explicitly cyclic settings. Understanding the inductive biases induced by training order is therefore important for understanding how language models acquire associations during training. More generally, our work supports the view that language models can, with caveats, be seen as sequence models over the training history.

References

[1] Kwangjun Ahn, Chulhee Yun, and Suvrit Sra. SGD with shuffling: Optimal rates without component convexity and large epoch requirements. *Advances in Neural Information Processing Systems*, 33:17526–17535, 2020.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision*, pages 139–154, 2018.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

[4] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[5] Zhiyuan Chen and Bing Liu. *Lifelong Machine Learning*, volume 1. Springer, 2018.

[6] Wikimedia Foundation. Wikimedia downloads, 2023. URL https://dumps.wikimedia.org.

[7] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023. URL https://arxiv.org/abs/2208.01066.

[8] Jonas Geiping, Micah Goldblum, Phil Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZBESeIUB5k.

[9] M Gurbuzbalaban, Asu Ozdaglar, and Pablo A Parrilo. Convergence rate of incremental gradient and incremental newton methods. *SIAM Journal on Optimization*, 29(4):2542–2565, 2019.

[10] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, 186, 10 2015. doi: 10.1007/s10107-019-01440-w.

[11] Jeff Z. HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 2315–2357. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/haochen21a.html.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska,

et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[13] Dmitrii Krasheninnikov, Egor Krasheninnikov, Bruno Mlodozeniec, Tegan Maharaj, and David Krueger. Implicit meta-learning may lead language models to trust more reliable sources. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[14] Dmitrii Krasheninnikov, Richard E. Turner, and David Krueger. Fresh in memory: Training-order recency is linearly encoded in language model activations. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Tn6famjSxN.

[15] Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Mario Šaško, Yacine Jernite, and Abhishek Thakur. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, 2021. URL https://aclanthology.org.

[16] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

[17] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

[18] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. In *Advances in Neural Information Processing Systems, NIPS 2020*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[19] Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33: 17309–17320, 2020.

[20] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[21] 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*, pages 2001–2010, 2017.

[22] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *7th International Conference on Learning Representations, ICLR*

2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1gTShAct7.

[23] Daniel A. Roberts. Sgd implicitly regularizes generalization error, 2021. URL https://arxiv.org/abs/2104.04874.

[24] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Itay Safran and Ohad Shamir. How good is SGD with random shuffling? In *Conference on Learning Theory*, pages 3250–3284. PMLR, 2020.

[26] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[27] Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[29] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 2012.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[31] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[32] Lijie Xu, Shuang Qiu, Binhang Yuan, Jiawei Jiang, Cedric Renggli, Shaoduo Gan, Kaan Kara, Guoliang Li, Ji Liu, Wentao Wu, et al. Stochastic gradient descent without full data shuffle. *arXiv preprint arXiv:2206.05830*, 2022.

[33] Yanlai Yang, Matt Jones, Michael C. Mozer, and Mengye Ren. Reawakening knowledge: Anticipatory recovery from catastrophic interference via structured training. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 82438–82464. Curran Associates, Inc., 2024. doi: 10.52202/079017-2621. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9628b97f74cd8f83ad12b631378b81e2-Paper-Conference.pdf.

[34] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[35] Liu Ziyin, Kangqiao Liu, Takashi Mori, and Masahito Ueda. Strength of minibatch noise in SGD. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uorVGbWV5sw.

Appendix A. Related Work

Structured training streams and ordered optimization. Prior work has studied optimization under ordered or reshuffled data streams, including random reshuffling SGD, without-replacement sampling, and cyclic training regimes, primarily focusing on convergence properties and optimization efficiency [1, 9, 10, 19, 25, 32]. Yang et al. [33] recently demonstrated that large language models trained in cyclic fine-tuning settings exhibit anticipatory recovery, whereby the loss on an upcoming document decreases before the document is revisited. Our work extends this line of research from repeated-document fine-tuning to bursty pre-training over classes, and argues that structured training streams induce a systematic sequential inductive bias. Rather than studying only convergence or forgetting, we focus on how temporal structure in the training order reshapes learned relationships between classes and representations.

Implicit biases of SGD and optimizer dynamics. A growing literature studies stochastic gradient descent not merely as an optimization algorithm, but as a source of implicit regularization and inductive bias. Prior analyses have connected SGD noise to flat minima, generalization, and gradient covariance structure [11, 23, 27, 35], but focused on i.i.d. batches. Implicit regularizers were found to aid generalization even when replaced by explicit regularizer in gradient descent [8]. Our theoretical analysis extends SGD implicit bias to bursty non-i.i.d. training streams.

Continual learning and catastrophic interference. Our work is also closely related to continual learning and catastrophic forgetting [17]. Neural networks trained sequentially on non-i.i.d. tasks often exhibit catastrophic interference, where learning new tasks degrades performance on previously learned ones. A large body of work has proposed mitigation strategies [2, 4, 5, 12, 21, 24, 26, 34]. Work in continual and online learning has also emphasized the role of task similarity, gradient interference, and transfer geometry in determining sequential learning dynamics [16, 18, 22]. These works study how gradients induced by different tasks can either conflict or facilitate one another, shaping forgetting and transfer. In this sense, our work connects catastrophic forgetting with optimization trajectories induced by structured training streams.

Emergent adaptation and implicit meta-learning. Recent work has interpreted Transformers as systems capable of implicit adaptation and meta-learning [13]. Large language models have been shown to exhibit emergent capabilities that appear only beyond certain scales [31]. Separately, recent work showed that transformer activations linearly encode recency information about when knowledge was acquired during training Krasheninnikov et al. [14], suggesting that optimization history may leave persistent traces in model representations. Our work complements these findings by studying transformers as partial sequence models over the training stream itself. Unlike standard in-context learning [7], where adaptation occurs within a forward pass over tokens, anticipation emerges through the optimization trajectory across training steps. We show that sufficiently large transformers can exploit predictable temporal structure in training order, achieving a substantial fraction of the anticipatory advantage obtainable by an ideal sequence model. These results suggest that scale can enable models to internalize higher-order statistical structure not only over token sequences, but also over the dynamics of the training process itself.

Relationship to anticipatory recovery [33]. While anticipatory recovery measures the loss trajectory of a repeated document, we instead wanted to measure adaptation to the structure of the training stream itself. This was motivated by our framing of anticipation as sequence model-like non-i.i.d.

behavior (see Proposition 2.2). In Figure 5, we show that our setup also reproduces anticipation when measured with the anticipatory recovery metric: the evaluation loss of a given class distribution starts decreasing before that class is revisited again.

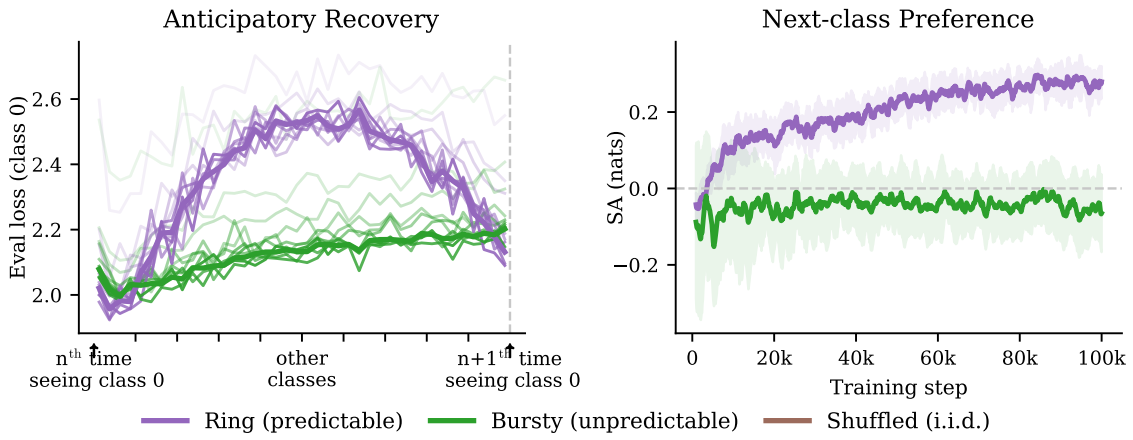

Figure 5: **Comparison of anticipatory recovery [33] and anticipation as next-class preference.** *(Left) Reproduced Anticipatory recovery over cycle progress for class* 0. The thick curves show eval loss, averaged over cycles in the last half of training. Lighter curves show 10% training windows, lighter means earlier. Much like in Yang et al. [33], the Ring configuration recovers most of its forgetting during the cycle *before seeing the class again*, while the Bursty setting doesn't. *(Right) Next-class Preference measured with Successor Advantage (SA) over time*, with ± 1 std intervals. Ring SA keeps increasing, while Bursty stays below zero. Values are averaged over 12 seeds.

Appendix B. Details on the experimental setup

B.1. Dataset

We use byte-level text from the `wikimedia/wikipedia` dataset [6] on Hugging Face Datasets [15], across ten languages: English, Dutch, German, French, Spanish, Portuguese, Italian, Polish, Czech, and Finnish. Each language constitutes one class. We use 1,000,000 training samples and 256 evaluation samples per language class. Text is chunked into contiguous segments of 8 tokens, except in the context length ablation, where we additionally use segments of 32, 64, 128 and 256 tokens. The default choice of sequence length was motivated by Proposition 2.2: which shows that, for a fixed number of classes, the amount of achievable anticipatory advantage in the training loss scales inversely with the context length. Short sequence lengths therefore provide a setting where anticipation is highly incentivized, allowing us to isolate the behavior easily. The model input additionally prepends a BOS token and appends an EOS token, yielding an default input length of 10. Text is encoded as UTF-8 bytes; each byte value is used directly as a token ID, with no subword tokenization. The vocabulary size is 258 (256 token values plus BOS and EOS).

In the Ring and Bursty setups, data is sampled from a Markov chain over classes, with transition probability $1/\tau$, with default $\tau = 2048$. This means that burst lengths are geometrically distributed, and on average, 2048 samples (organized in batches) are seen from a class before a transition.

B.2. Model architectures

We train GPT-2 style transformers [30] of several scales. All models share the same byte-level vocabulary of size 258 and input length 10 (excluding the "Empirical limits of anticipation" experiment and the context length ablation). Architecture details are given in Table 1.

Parameters	**Hidden dim** (H)	**Layers** (L)	**Heads** (A)	**Intermediate dim** (I)
0.19M	48	6	2	144
1.31M	128	6	4	384
5.2M	256	6	8	768
20.6M	512	6	8	1536
51.9M	576	12	8	1728

Table 1: **Model configurations.** All models use the same byte-level vocabulary (size 258) and input length 10. Parameter counts include embedding layers.

B.3. Training hyperparameters

All experiments use AdamW with weight decay 0.1, AMP mixed precision, and a warmup of 1,000 steps. "Gradient steps per class" refers to the expected number of gradient updates taken while the Ring chain remains in a single class, equal to τ/B where τ is the mean burst length and B is the batch size. Hyperparameters for the default configuration and each ablation (except for the optimizer ablation, whose hyperparamaters are reported in Table 5 and discussed in Section D.5) are summarised in Table 2.

B.4. Training protocols

We train under three protocols. See Figure 1 (left) for a visual explanation of these orderings.

- **Ring.** Training examples are drawn from a 10-class Markov chain. Classes are arranged in a fixed cyclic order, which, unless stated otherwise, depends on the random seed used for the experiment. At each step the chain stays in the current class with probability $p_{\text{stay}} = 1 - 1/\tau$, giving geometrically distributed burst lengths with mean τ, where τ equals the batch size times the number of gradient steps per class. When a transition occurs, the chain deterministically moves to the next class in the ring.
- **Bursty.** The chain has the same geometric burst structure as Ring (same τ and p_{stay}), but transitions are drawn uniformly over all other classes rather than following a fixed cyclic order. This condition controls for the burst structure while removing any predictable successor identity.
- **Shuffled.** Training examples are drawn uniformly at random from the mixture of all classes, with no temporal structure. This is the standard i.i.d. training baseline.

Experiment	Batch size	Context len	Grad steps	Total steps	Optimizer	LR	Wd
Default (Figure 1)	64	8	32	100k	AdamW	5×10^{-5}	0.1
Gradient steps ablation (Figure 10)	64	8	8 16 32 64 128	100k	AdamW	5×10^{-5}	0.1
Context length ablation (Figure 11)	64	8 32 128 256	32	100k	AdamW	5×10^{-5}, 1×10^{-4} 2×10^{-4}, 3×10^{-4}	0.1
Batch size ablation (Figure 12)	4 16 64 256	8	32	100k	AdamW	1.25×10^{-5}, 2.5×10^{-5}, 5×10^{-5}, 1×10^{-4}	0.1
Model size ablation (Figure 13)	64	8	32	100k	AdamW	1×10^{-4}, 1×10^{-4}, 5×10^{-5}, 5×10^{-5}, 5×10^{-5}	0.1

Table 2: **Training hyperparameters for each experiment.** Grad steps should be understood as *expected* number of grad steps as burst lengths are geometrically distributed. All runs use weight decay 0.1 and a 1,000-step linear warmup. In multi-row ablation cells, LR and batch size values correspond to the listed context lengths or model sizes top to bottom.

B.5. Evaluation metrics

Successor Advantage. Successor Advantage is defined in Definition 2.1 as:

$$SA_t = \frac{1}{|\{x : x \notin c_{\text{current}} \cup c_{\text{next}}\}|} \sum_{x \notin c_{\text{current}} \cup c_{\text{next}}} \ell(x) - \frac{1}{|c_{\text{next}}|} \sum_{x \in c_{\text{next}}} \ell(x).$$

In practice, we compute a *corrected variant*, which removes the bias of some successor classes being inherently easier to predict than others, which could cause accidental positive SA in Ring setups. Since Bursty doesn't have deterministic successor, shuffle-correction is not necessary. The corrected SA for Ring is the SA relative to a paired Shuffled model: $SA_{\text{corrected},t} = SA_{\text{Ring},t} - SA_{\text{Shuffled},t}$. For both Ring and Shuffled, SA computation uses the Ring's successor and current labels. The paired Shuffled model is trained on a uniformly random permutation of the identical token stream seen by the Ring model, so both models see the same multiset of examples. Only the temporal ordering differs. This ensures that any "accidental" SA in the shuffled model, arising from inherent differences in class difficulty or vocabulary rather than sequential structure, is removed. In practice, the shuffle correction only marginally changes the SA. For Bursty, raw SA is reported, as All SA values are averaged over the last 25% of training steps.

Appendix C. Proofs

C.1. Theoretical limit of anticipatory advantage

Setup. Let $x = (x_1, x_2, ..., x_M)$ denote a data sequence, where each token x_i is drawn from a fixed token dictionary $\mathcal{V}$. Let D_{train} denote a dataset of N such samples drawn i.i.d. from some mixture distribution over classes $c : p(x) = \sum_{c \in C} p(x \mid c)p(c)$. For simplicity, assume that classes are discrete, and have equal marginal probability. Autoregressive language models approximate $p(x)$ by next-token prediction, by fitting a distribution $p_M(x_{i+1} \mid x_{1:i}; \theta)$, parametrized by θ. Crucially, the architecture is memoriless: it does not explicitly model the training order $x^1, x^2, \dots x^{|D|}$.

Sequence models. Anticipation is a property of *sequence models*, which approximate $p(x^i \mid x^1, x^2, \dots, x^{i-1})$. In the setup we consider, the training order cycles through the classes c, hence $p(x^i \mid x^1, x^2, \dots, x^{i-1}) = p(x^i \mid c : x_i \in c)$.

Proposition C.1 *Let C denote the set of classes and M the sequence length. The best possible average loss improvement over shuffled training via anticipation is $\log |C|/M$. This is achieved by sequence models explicitly modeling class order, i.e. $p_{\text{sequence model}}(x) = p(x \mid c : x \in c)$.*

Proof The per-token cross-entropy loss for a model $q(x)$ is

$$\mathbb{E}_{x \sim p(x)}\Big[-\frac{1}{M} \sum_{i=1}^{M} \log q(x_i | x_1, x_2, \dots, x_{i-1}) \Big] = \mathbb{E}_{x \sim p(x)}\Big[-\frac{1}{M} \log q(x) \Big].$$

Training on shuffled batches targets $p(x)$, hence the best achievable cross-entropy loss is $\ell_{\text{shuffled}}(x) = \mathbb{E}_{x \sim p(x)}\Big[-\frac{1}{M} \log p(x) \Big]$. Perfect sequence anticipation means that the model learns $p(x|c)$, wherever $x \in c$. Hence the best achievable loss is $\ell_{\text{anticipatory}}(x) = \mathbb{E}_{c, x \sim p(c,x)}\Big[-\frac{1}{M} \log p(x \mid c) \Big]$. From Bayes's rule, we have

$$p(x \mid c) = \frac{p(x)p(c \mid x)}{p(c)} \approx \frac{p(x)}{p(c)},$$

because we assumed that c is identifiable from x, hence $p(c|x) \approx 1$. Hence the best possible loss improvement is

$$\begin{aligned} \ell_{\text{shuffled}}(x) - \ell_{\text{anticipatory}}(x) &= \mathbb{E}_{c, x \sim p(c,x)}\Big[\frac{1}{M} \Big[-\log p(x) + \log p(x) - \log p(c) \Big] \Big] \\ &\approx \frac{\mathbb{E}_{c \sim p(c)}[-\log p(c)]}{M}. \end{aligned}$$

For $p(c) = \frac{1}{|C|}$, this simplifies to $\frac{\log |C|}{M}$. ■

C.2. Anticipation as the implicit bias of bursty mini-batch training

In this section, we characterize the implicit bias of SGD during bursty training, by extending the analysis of Roberts [23]. We study two sequential SGD updates on batches B_{current} and B_{next}, starting from θ_0, and ask how the construction of mini-batches shapes optimization trajectory compared to full-batch gradient descent. For readability, we will use the shorthands B_c and B_n. We are interested in the below two settings:

- Shuffled training: B_c and B_n are i.i.d. draws from the data mixture $p(x)$.
- Bursty training: B_c and B_n each only contain samples from a single class c_{current} and c_{next}, respectively.

Roberts [23]'s analysis assumes the shuffled setting. Both settings take two sequential gradient steps of size η on batches of size $N/2$, and we compare this two-step SGD trajectory to the equivalent two full-batch GD steps on $B_c \cup B_n$, isolating the effect of sequential updating.

Notation Let $p(x)$ be a data-generating distribution, which can be written as a mixture $p(x) = \sum_{c \in C} p(x \mid c)p(c)$. Let $\ell(\theta, x)$ be a twice continuously differentiable loss function with parameter vector $\theta \in \mathbb{R}^d$, and define the per-example gradient and Hessian:

$$g_i(\theta, x) = \partial \ell(\theta, x)/\partial \theta_i, \qquad h_{ij}(\theta, x) = \partial_i \partial_j \ell(\theta, x)/\partial \theta_i \partial \theta_j.$$

Under $p(x)$, define the full-data gradient mean and covariance

$$G_i^{(\text{full})}(\theta) = \mathbb{E}_{x \sim p}[g_i(\theta, x)], \qquad \Sigma_{ij}^{(\text{full})}(\theta) = \mathrm{Cov}_{x \sim p}(g_i, g_j),$$

and the per-class gradient means and covariances:

$$G_i^{(B_n)} \equiv \mathbb{E}_{x \sim p_c}[g_i(\theta, x)], \qquad G_i^{(\text{full})} = \sum_{c \in C} p(c) G_i^{(c)}$$

$$\Sigma_{ij}^{(B_n)} \equiv \mathrm{Cov}_{p_c}(g_i, g_j), \qquad \Sigma^{(\text{full})} = \Sigma_{\text{within}} + \Sigma_{\text{between}}$$

Further, for batch B, define the sample mean gradient and Hessian as

$$g_i^{(B_c)}(\theta) = \frac{1}{|B|} \sum_{x \in B} g_i(\theta, x), \qquad h_{ij}^{(B_c)}(\theta) = \frac{1}{|B|} \sum_{x \in B} h_{ij}(\theta, x).$$

Let the change in parameter θ^i after updates on two SGD batches B_c, B_n be denoted as $\delta\theta_2^{i(B_c, B_n)}$, and the change after two gradient descent updates on the full data $B_c \cup B_n$ be $\delta\theta_2^{i(B_c \cup B_n, B_c \cup B_n)}$. Then the difference in gradient δg_i satisfies:

$$\delta\theta_2^{i(B_c \cup B_n, B_c \cup B_n)} - \delta\theta_2^{i(B_c, B_n)} = -\eta\, \delta g_i.$$

We rely on the following regularity assumptions.

Assumptions

(i) **Finite second moments**: $\mathrm{tr}\, \Sigma < \infty$;

(ii) **Independence of minibatches** B_c and B_n.

(iii) **Second-order smoothness.** The loss $\ell(\theta, x)$ is twice continuously differentiable with respect to θ, and its Hessian is locally Lipschitz in a neighborhood of the optimization trajectory. This ensures that the Taylor expansions used below are valid up to $O(\eta^2)$ remainder terms;

(iv) **Differentiation under the expectation.** For every class distribution considered, the functions $g_i(\theta, x)$, $h_{ij}(\theta, x)$, and the products $g_i(\theta, x)g_j(\theta, x)$ are differentiable in a neighborhood of the current parameter θ, and their derivatives are dominated by integrable functions independent of θ. This ensures, by dominated convergence, that derivatives may be exchanged with expectations.

(v) **Approximately equal within-class gradient variance:** $\operatorname{tr}\Sigma^{(B_c)} \approx \operatorname{tr}\Sigma^{(B_n)}$

We note that assumption (v) is not integral, and is only made to simplify the final formula. We first state the shuffled-SGD implicit bias result from Roberts [23], and replicate the proof in our notation.

Theorem C.2 (Implicit Bias of Shuffled SGD [23]) *Let $B_{current}$ and B_{next} be two independent minibatches of size $N/2$, drawn i.i.d. from $p(x)$. Then, to leading order in η,*

$$\mathbb{E}[\delta g_i^{shuffled}] = -\frac{\eta}{2N}\,\partial_i \operatorname{tr}\Sigma(\theta)$$

where the expectation is taken over the randomness in the minibatches B_c and B_n. Equivalently, the expected SGD update differs from the corresponding GD update by a drift term proportional to the gradient of the trace of the gradient covariance:

$$\mathbb{E}[\delta g^{shuffled}] = -\frac{\eta}{2N}\,\nabla_\theta \operatorname{tr}\Sigma(\theta).$$

The proof requires the following Lemma that our bursty result will rely on as well.

Lemma C.3 *When comparing GD to SGD for any two batches B_c and B_n, we have*

$$\delta g_i = -\frac{\eta}{8}\left[\partial_i \|g^{(B_c)} - g^{(B_n)}\|^2 + 4\big(g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\big)\right] + O(\eta^2).$$

Proof Let the initial parameter be θ_0. As in Section 3 of Roberts [23], consider two subsequent SGD updates with minibatches B_c and B_n, each of size $N/2$:

$$\theta_1^{i(B_c)} = \theta_0^i - \eta g_i^{(B_c)}(\theta_0), \qquad \theta_2^{i(B_c,B_n)} = \theta_1^{i(B_c)} - \eta g_i^{(B_n)}(\theta_1^{(B_c)}). \tag{1}$$

We compare this to the equivalent of two full-data GD updates using the union batch $B_c \cup B_n$ of size N:

$$\theta_1^{i(B_c\cup B_n)} = \theta_0^i - \eta g_i^{(B_c\cup B_n)}(\theta_0), \qquad \theta_2^{i(B_c\cup B_n, B_c\cup B_n)} = \theta_1^{i(B_c\cup B_n)} - \eta g_i^{(B_c\cup B_n)}(\theta_1^{i(B_c\cup B_n)}). \tag{2}$$

We now expand both trajectories to second order in η. Substituting the first equation of (1) into the second gives

$$\theta_2^{i(B_c,B_n)} = \theta_0^i - \eta g_i^{(B_c)}(\theta_0) - \eta g_i^{(B_n)}(\theta_{1(B_c)}).$$

A one-step Taylor expansion of the last gradient around θ_0 gives:

$$g_i^{(B_n)}(\theta_1^{(B_c)}) = g_i^{(B_n)}(\theta_0) + \partial_j g_i^{(B_n)}(\theta_0)\,(\theta_1^{j(B_c)} - \theta_0^j) + O(\eta^2).$$

Since $\theta_1^{j(B_c)} - \theta_0^j \approx -\eta g_j^{(B_c)}(\theta_0)$ and $\partial_j g_i^{(B_n)} = h_{ij}^{(B_n)}$ for small enough η, this becomes

$$g_i^{(B_n)}(\theta_1^{(B_c)}) = g_i^{(B_n)}(\theta_0) - \eta h_{ij}^{(B_n)}(\theta_0) g_j^{(B_c)}(\theta_0) + O(\eta^2).$$

Hence

$$\theta_2^{i(B_c,B_n)} - \theta_0^i = -\eta\big(g_i^{(B_c)} + g_i^{(B_n)}\big) + \eta^2 h_{ij}^{(B_n)} g_j^{(B_c)} + O(\eta^3), \tag{3}$$

where from now on, all gradients and Hessians are evaluated at θ_0 unless explicitly stated otherwise. Now, we expand of the one-step GD iterate on $B_c \cup B_n$. Similarly,

$$\theta_2^{i(B_c\cup B_n,B_c\cup B_n)} = \theta_0^i - \eta g_i^{(B_c\cup B_n)}(\theta_0) - \eta g_i^{(B_c\cup B_n)}(\theta_1^{(B_c\cup B_n)}).$$

Taylor expanding the second gradient around θ_0 gives

$$g_i^{(B_c\cup B_n)}(\theta_1^{(B_c\cup B_n)}) = g_i^{(B_c\cup B_n)} - \eta h_{ij}^{(B_c\cup B_n)} g_j^{(B_c\cup B_n)} + O(\eta^2),$$

and therefore

$$\theta_2^{i(B_c\cup B_n,B_c\cup B_n)} - \theta_0^i = -2\eta g_i^{(B_c\cup B_n)} + \eta^2 h_{ij}^{(B_c\cup B_n)} g_j^{(B_c\cup B_n)} + O(\eta^3). \tag{4}$$

Because $|B_c| = |B_n| = N/2$, the union-batch gradient and Hessian are simple averages:

$$g_i^{(B_c\cup B_n)} = \frac{1}{2}\big(g_i^{(B_c)} + g_i^{(B_n)}\big), \qquad h_{ij}^{(B_c\cup B_n)} = \frac{1}{2}\big(h_{ij}^{(B_c)} + h_{ij}^{(B_n)}\big).$$

Substituting these identities into (4) yields

$$\theta_2^{i(B_c\cup B_n,B_c\cup B_n)} - \theta_0^i = -\eta\big(g_i^{(B_c)} + g_i^{(B_n)}\big) + \frac{\eta^2}{4}\big(h_{ij}^{(B_c)} + h_{ij}^{(B_n)}\big)\big(g_j^{(B_c)} + g_j^{(B_n)}\big) + O(\eta^3). \tag{5}$$

Since $\big(\theta_2^{i(B_c\cup B_n,B_c\cup B_n)} - \theta_0^i\big) - \big(\theta_2^{i(B,C)} - \theta_0^i\big) \equiv -\eta\,\delta g_i$, Subtracting (3) from (5), the $O(\eta)$ terms cancel, and we obtain

$$-\eta\,\delta g_i = \eta^2\left[\frac{1}{4}(h_{ij}^{(B_c)} + h_{ij}^{(B_n)})(g_j^{(B_c)} + g_j^{(B_n)}) - h_{ij}^{(B_n)} g_j^{(B_c)}\right] + O(\eta^3). \tag{6}$$

Expanding the bracket and using the identity $h_{ij}g_j = \partial_i\big(\frac{1}{2}\|g\|^2\big)$, separately to the batch gradients $g^{(B_c)}$ and $g^{(B_n)}$, we get

$$\delta g_i = -\frac{\eta}{8}\partial_i\Big(\|g^{(B_c)}\|^2 + \|g^{(B_n)}\|^2\Big) - \frac{\eta}{4}\Big(h_{ij}^{(B_c)} g_j^{(B_n)} - 3h_{ij}^{(B_n)} g_j^{(B_c)}\Big) + O(\eta^2). \tag{7}$$

Adding and subtracting $2g^{(B_c)}\cdot g^{(B_n)}$ inside the derivative gives:

$$\|g^{(B_c)}\|^2 + \|g^{(B_n)}\|^2 = \|g^{(B_c)} - g^{(B_n)}\|^2 + 2g^{(B_c)}\cdot g^{(B_n)}.$$

Since $\partial_i\big(g^{(B_c)}\cdot g^{(B_n)}\big) = h_{ij}^{(B_c)} g_j^{(B_n)} + h_{ij}^{(B_n)} g_j^{(B_c)}$, equation (7) becomes

$$\begin{aligned}\delta g_i &= -\frac{\eta}{8}\partial_i\|g^{(B_c)} - g^{(B_n)}\|^2 - \frac{\eta}{4}\partial_i\big(g^{(B_c)}\cdot g^{(B_n)}\big) - \frac{\eta}{4}\Big(h_{ij}^{(B_c)} g_j^{(B_n)} - 3h_{ij}^{(B_n)} g_j^{(B_c)}\Big) + O(\eta^2)\\ &= -\frac{\eta}{8}\partial_i\|g^{(B_c)} - g^{(B_n)}\|^2 - \frac{\eta}{2}\Big(g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\Big) + O(\eta^2).\end{aligned} \tag{8}$$

Equivalently,

$$\delta g_i = -\frac{\eta}{8}\left[\partial_i \|g^{(B_c)} - g^{(B_n)}\|^2 + 4\big(g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\big)\right] + O(\eta^2). \tag{9}$$

■

We now prove the implicit bias result for *shuffled* SGD.

Proof of Theorem C.2. B_c and B_n are i.i.d. draws from the full mixture $p(x)$. For a minibatch of size $N/2$, the batch gradients satisfy:

$$\mathbb{E}[g_i^{(B_c)}] = G_i^{(\text{full})}, \qquad \text{Cov}(g_i^{(B_c)}, g_j^{(B_c)}) = \frac{2}{N}\Sigma_{ij}^{(\text{full})},$$

and the same for $g^{(B_n)}$. Since in the shuffled case, B_c and B_n are independent and identically distributed,

$$\mathbb{E}[g^{(B_c)} \cdot g^{(B_n)}] = \mathbb{E}[g^{(B_c)}] \cdot \mathbb{E}[g^{(B_n)}] = \|G^{(\text{full})}\|^2.$$

Also,

$$\mathbb{E}\Big[\|g^{(B_c)}\|^2\Big] = \|G^{(\text{full})}\|^2 + \frac{2}{N}\,\text{tr}\,\Sigma^{(\text{full})}.$$

Hence

$$\mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] = \mathbb{E}\Big[\|g^{(B_c)}\|^2\Big] + \mathbb{E}\Big[\|g^{(B_n)}\|^2\Big] - 2\mathbb{E}\Big[g^{(B_c)} \cdot g^{(B_n)}\Big] = \frac{4}{N}\,\text{tr}\,\Sigma^{(\text{full})}.$$

For the antisymmetric term in (9), independence and identical distribution imply

$$\mathbb{E}\Big[g_j^{(B_n)} h_{ij}^{(B_c)}\Big] = \mathbb{E}[g_j^{(B_n)}]\mathbb{E}[h_{ij}^{(B_c)}] = G_j^{(\text{full})} H_{ij}^{(\text{full})},$$

and similarly $\mathbb{E}\Big[g_j^{(B_c)} h_{ij}^{(B_n)}\Big] = G_j^{(\text{full})} H_{ij}^{(\text{full})}$, so $\mathbb{E}\Big[g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\Big] = 0$. Taking expectation in (9) and commuting ∂_i with expectation (justified by assumption (v)), we conclude

$$\begin{aligned}
\mathbb{E}[\delta g_i^{\text{shuffled}}] &= -\frac{\eta}{8}\,\partial_i \mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] + O(\eta^2) \\
&= -\frac{\eta}{8}\,\partial_i\left(\frac{4}{N}\,\text{tr}\,\Sigma^{(\text{full})}\right) + O(\eta^2) \\
&= -\frac{\eta}{2N}\,\partial_i\,\text{tr}\,\Sigma^{(\text{full})} + O(\eta^2).
\end{aligned} \tag{10}$$

This corresponds to an isotropic bias toward regions of low gradient variance across the full dataset.■

We can now prove our theorem for bursty training.

Theorem C.4 (Implicit bias of bursty training) *Let $B_c \overset{i.i.d.}{\sim} p_{current}(x)$ and $B_n \overset{i.i.d.}{\sim} p_{next}(x)$ be two minibatches of size $N/2$. Assume that $\text{tr}\,\Sigma^{(c)} \approx \text{tr}\,\Sigma^{(n)}$. Then, to leading order in η,*

$$\mathbb{E}[\delta g_i^{bursty}] = -\frac{\eta}{2N}\,\partial_i\,\text{tr}\,\Sigma_{within} - \frac{\eta}{8}\,\partial_i\|G^{(c)} - G^{(n)}\|^2 - \frac{\eta}{2}\Big(G_j^{(n)} H_{ij}^{(c)} - G_j^{(c)} H_{ij}^{(n)}\Big),$$

where the expectation is taken over the randomness in the minibatches B_c and B_n.

Proof We start from Theorem C.3:

$$\mathbb{E}[\delta g_i] = -\frac{\eta}{8}\,\partial_i\,\mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] - \frac{\eta}{2}\,\mathbb{E}\Big[g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\Big] + O(\eta^2). \tag{11}$$

In the bursty setting, the two expectations in (11) are no longer symmetric under $B_c \leftrightarrow B_n$, because the two batches are drawn from different class-conditional distributions. Recall the class-conditional means and covariances

$$G_i^{(c)} = \mathbb{E}_{x\sim p_{\text{current}}}[g_i(\theta, x)], \qquad \Sigma_{ij}^{(c)} = \mathrm{Cov}_{x\sim p_{\text{current}}}(g_i, g_j),$$

and similarly for class c_{next}. Since B_c is a minibatch of size $N/2$ drawn i.i.d. from p_{current}, its batch gradient satisfies

$$\mathbb{E}[g_i^{(B_c)}] = G_i^{(c)}, \qquad \mathrm{Cov}(g_i^{(B_c)}, g_j^{(B_c)}) = \frac{2}{N}\Sigma_{ij}^{(c)}.$$

Hence

$$\mathbb{E}\Big[\|g^{(B_c)}\|^2\Big] = \|G^{(c)}\|^2 + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(c)}. \tag{12}$$

Similarly,

$$\mathbb{E}\Big[\|g^{(B_n)}\|^2\Big] = \|G^{(n)}\|^2 + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(n)}. \tag{13}$$

Because B_c and B_n are independent,

$$\mathbb{E}[g^{(B_c)} \cdot g^{(B_n)}] = \mathbb{E}[g^{(B_c)}] \cdot \mathbb{E}[g^{(B_n)}] = G^{(c)} \cdot G^{(n)}. \tag{14}$$

Therefore,

$$\begin{aligned}
\mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] &= \mathbb{E}\Big[\|g^{(B_c)}\|^2\Big] + \mathbb{E}\Big[\|g^{(B_n)}\|^2\Big] - 2\,\mathbb{E}[g^{(B_c)} \cdot g^{(B_n)}] \\
&= \|G^{(c)}\|^2 + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(c)} + \|G^{(n)}\|^2 + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(n)} - 2\,G^{(c)} \cdot G^{(n)} \\
&= \|G^{(c)} - G^{(n)}\|^2 + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(c)} + \frac{2}{N}\,\mathrm{tr}\,\Sigma^{(n)}.
\end{aligned} \tag{15}$$

Due to the assumption $\mathrm{tr}\,\Sigma^{(c)} \approx \mathrm{tr}\,\Sigma^{(n)} \approx \mathrm{tr}\,\Sigma_{\text{within}}$, (15) becomes

$$\mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] = \|G^{(c)} - G^{(n)}\|^2 + \frac{4}{N}\,\mathrm{tr}\,\Sigma_{\text{within}}. \tag{16}$$

Substituting (16) into the first term of (11) gives

$$\begin{aligned}
-\frac{\eta}{8}\partial_i\,\mathbb{E}\Big[\|g^{(B_c)} - g^{(B_n)}\|^2\Big] &= -\frac{\eta}{8}\partial_i\left(\|G^{(c)} - G^{(n)}\|^2 + \frac{4}{N}\,\mathrm{tr}\,\Sigma_{\text{within}}\right) \\
&= -\frac{\eta}{8}\partial_i\|G^{(c)} - G^{(n)}\|^2 - \frac{\eta}{2N}\partial_i\,\mathrm{tr}\,\Sigma_{\text{within}}.
\end{aligned} \tag{17}$$

Since B_c and B_n are independent, and $g^{(B_n)}$ depends only on B_n while $h^{(B_c)}$ depends only on B_c, we may factor expectations:

$$\mathbb{E}[g_j^{(B_n)} h_{ij}^{(B_c)}] = \mathbb{E}[g_j^{(B_n)}]\,\mathbb{E}[h_{ij}^{(B_c)}] = G_j^{(n)} H_{ij}^{(c)}. \tag{18}$$

Therefore

$$\mathbb{E}\Big[g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\Big] = G_j^{(n)} H_{ij}^{(c)} - G_j^{(c)} H_{ij}^{(n)}. \tag{19}$$

Substituting (19) into the second term of (11) yields

$$-\frac{\eta}{2}\,\mathbb{E}\Big[g_j^{(B_n)} h_{ij}^{(B_c)} - g_j^{(B_c)} h_{ij}^{(B_n)}\Big] = -\frac{\eta}{2}\Big(G_j^{(n)} H_{ij}^{(c)} - G_j^{(c)} H_{ij}^{(n)}\Big). \tag{20}$$

Combining (17) and (20) in (11), we obtain

$$\mathbb{E}[\delta g_i^{\text{bursty}}] = -\frac{\eta}{2N}\,\partial_i \operatorname{tr}\Sigma_{\text{within}} - \frac{\eta}{8}\,\partial_i \|G^{(c)} - G^{(n)}\|^2 - \frac{\eta}{2}\Big(G_j^{(n)} H_{ij}^{(c)} - G_j^{(c)} H_{ij}^{(n)}\Big) + O(\eta^2).$$

■

Interpretation. Three distinct effects appear in the bursty setting.

1. **Variance regularization**: the same gradient noise reduction mechanism as in shuffled SGD, but restricted to within-class variance
2. **Alignment pressure**: a symmetric force that pushes $G^{(c)}(\theta)$ and $G^{(n)}(\theta)$ toward each other.
3. **Curvature interaction**: an antisymmetric term capturing how a gradient step on one class modifies the landscape encountered by the next.

The main candidate for driving anticipation is Term 2:$-\frac{\eta}{8}\,\partial_i \|G^{(c)} - G^{(n)}\|^2$. Under cyclic training, that is, repeatedly cycling through $c_{\text{current}}, c_{\text{next}}$ and other classes in fixed order, Term 2 is repeatedly applied and encourages alignment of class-conditional gradients. When $G^{(c)}(\theta) \approx G^{(n)}(\theta)$, updates from one class can become beneficial for the next, producing anticipatory recovery. However, the remaining two terms may also impact the overall effect. For example, Term 3 introduces curvature-dependent interference, which may reinforce or oppose alignment. The relationship of Term 2 and Term 3, and their impact on anticipation is studied in Section D.5.

Appendix D. Further results and analysis

D.1. Details on the *"empirical limit of anticipation"* experiment

In measuring agreement with a sequence model in Section 2, our goal was to test to what extent our trained Transformer M behaves like a sequence model. The only predictable information from the training history was the class of the next batch. Therefore, a perfect sequence model equals $p(x \mid c : x \in c)$, which corresponds to an average train loss of the conditional entropy $H(x|c)$. However, failing to reach this limit can happen due to reasons different from the core mechanism of anticipation; for example, limited model expressivity or limited training duration. Hence, in this experiment, we estimate the maximal anticipation achievable *by the given model and the used training duration*, giving a model- and experiment-dependent empirical limit of achievable anticipation.

Experiment details. We estimate this empirical limit of anticipation by artificially giving class information to the model in-context, creating an in-context oracle baseline. We augment our training and evaluation samples by prepending the (correct) class information, so the sample becomes `BOS @@C@@ {class_id} || {original_data} EOS`, where `class_id` is the numeric class index (0–9), encoded as UTF-8 bytes, `||` are separator tokens, and `{original_data}` is the original data sample. This format was chosen to make such patterns unlikely to accidentally appear in the data. We compute the training and evaluation losses via masking the class cue and separator tokens out, hence the model is not trained to predict the cue and separator. This ensures that the model is trained to match the class index to the data content. At evaluation time, this acts as an implicit conditioning on the correct class. Other than the above changes, our experimental configuration is identical to the default experiment (see Table 2).

Results. The results are shown in Figure 2. The theoretical limit was computed as 1.6913 (see Proposition 2.2), and the empirical limit at 100k steps is estimated as 1.7682. Ring achieves a training loss of 1.8938 ± 0.0175 at 100k, beating shuffled with training loss 1.9791. This corresponds to realizing 29.63% of the achievable theoretical advantage, and 40.43% of the achievable empirical advantage.

D.2. Geometric trajectory of model weights

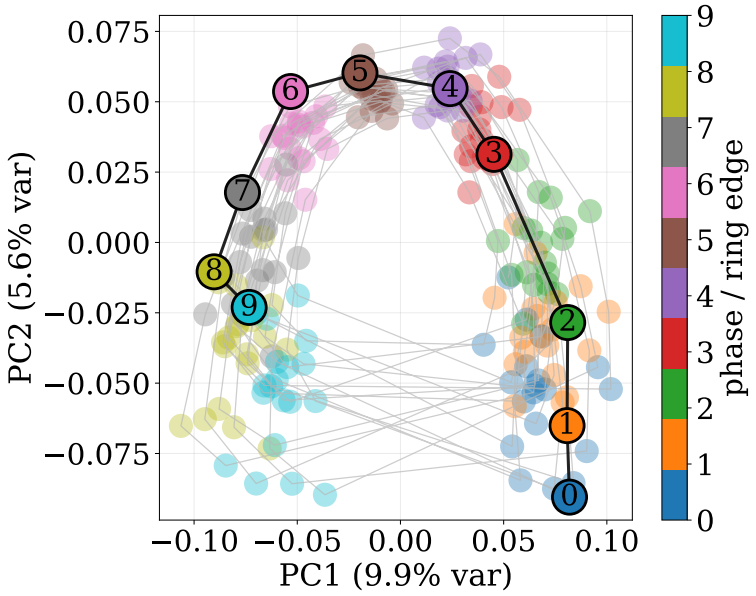

Figure 6: **Geometric structure of model weights.** We projected the flattened model weights recorded just before class transitions. 20 full cycles (200 transitions) are recorded, the last one is highlighted. Before PCA, the mean weight vector in each Ring loop is substracted to remove slow training drift. This removes most of the variance in data. The cyclic organization indicates that model weights enter a reproducible cyclic state across the ordered class transitions.

We visualize the trajectory of model weights immediately before class transitions in a Ring training run. We use a 0.19M-parameter model with sequence length 8, 32 gradient steps per class, batch size 64, and learning rate 5e-5, trained for 100k steps. In this experiment, we enforce fixed burst lengths for simplicity. We save pre-transition checkpoints at the last 200 class transitions. For each checkpoint, we flatten the model parameters into a single weight vector. To remove slow training drift, we group checkpoints into complete 10-transition Ring loops and subtract the mean vector of each loop from its corresponding checkpoints. We then apply PCA to these "centered" weight vectors.

Figure 6 shows the first two principal components. The colored clusters correspond to the transition phase, or Ring edge. Their circular organization shows that the model weights enter a reproducible cyclic state across the ordered class transitions, rather than merely drifting monotonically during training. This geometry is is consistent with gradient alignment towards successor classes.

D.3. Anticipation in second-order setups

We test whether anticipation emerges beyond first order Markov chains.

Experiment details. We implement a second-order transition rule over Wikipedia language classes. Instead of the next class being determined by the current class alone, it is determined by the ordered pair (previous class, current class). The transition table is a Latin square. We test a 5 and 6-class version, which have 25 and 36 states, respectively. The Latin square configurations are reported in Figure 7. For any fixed current class, each possible next class appears exactly once across predecessor classes. This makes the first-order marginal uninformative: knowing only the current class gives a uniform distribution over next classes. A model can only do better than chance if training has left it with usable information about the predecessor that led into the current burst. The tables are chosen to form a full orbit over joint states, so the deterministic class-pair dynamics cycle through all class pairs before repeating, giving a second-order equivalent of the original Ring configuration.

During a burst, the model sees batches from the current class. Contrary to the first-order experiments, here we enforce switches to happen deterministically, after a set burst length/number of gradient steps on a class (e.g. 32). When a switch happens, the tuple (previous class, current class) determines the next class, and after the switch, the latent state updates to (current class, next class). Self-successors, i.e., when next class is the same as the current class are possible, but these are excluded from Successor Advantage computation because no visible class switch occurs.

SA computation. Similarly to Section B.5, we use a corrected version of the Successor Advantage metric. From the raw SA, we substract a Bursty control SA. The Bursty control is a training stream that keeps the same burst structure but removes the predictable transition rule. This removes any anticipation effect due to factors not intrinsic to anticipation, such as class difficulty, current-class adaptation, or consequences of long same-class bursts. SA values are averaged over 16 seeds.

Results. Section D.3 reports the conducted experiments and corresponding SA results. The values are more modest than in the first-order setting. However, anticipation emerges in both 5-class and 6-class setups, and the ideal number of gradient steps (32) matches the first-order setup.

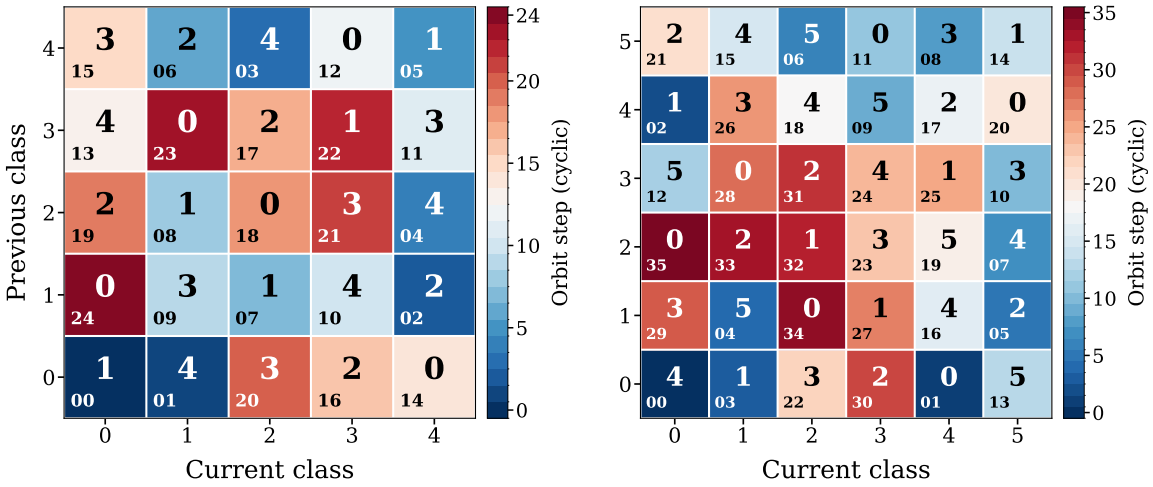

Figure 7: **The latin squared 2nd order transition function.** *(Left) 5-class configuration. (Right) 6-class configuration.* In both matrices, x and y axis labels show the individual state index. Colors and small-sized numbers in each cell show the index of state pairs (previous, current). Large-sized digits denote the next class from that cell pair. The orbit starts from the bottom left.

Number of classes	Gradient steps until switch	SA $\pm$ 1 std
5	16	-0.0053 ± 0.0045
5	**32**	$\mathbf{+0.0412 \pm 0.0081}$
5	64	$+0.0076 \pm 0.0105$
6	16	-0.0065 ± 0.0008
6	**32**	$\mathbf{+0.0152 \pm 0.0028}$
6	64	-0.0048 ± 0.0034

Table 3: **SA in the second-order setting.** Results are computed from 16 seeds per experiment. Both the 5 and 6-class setups display anticipation.

D.4. Anticipation in image modeling

We include MNIST experiments as an existence result showing that successor class anticipation is not specific to byte-level language modeling. Yang et al. [33] showed that anticipatory recovery arises in individual image-cycling.

Experiment details. We train autoregressive image models on MNIST. Each digit class is treated as one class in the Markov chain. Images are downsampled to 14×14, quantized to 8 grayscale levels, and flattened into a sequence of 196 image tokens. We prepend a BOS token and append an EOS token, giving sequence length 198 and vocabulary size 10. The model never receives the digit label as input or target, labels are used only to construct the class-structured training stream and to compute per-class evaluation losses.

The training protocols are the same as in Section B. In the Ring condition, digit classes are arranged in a seed-dependent cyclic order, and a transition deterministically moves to the next class in the ring. In the Bursty condition, burst lengths are matched, but the successor class is uninformative.

In the Shuffled condition, the same image stream is randomly permuted, removing temporal structure while preserving the multiset of examples.

We use a GPT-style Transformer [30] with hidden dimension 128, 4 layers, 4 attention heads, and intermediate dimension 512, for approximately 1.05M parameters. Models are trained for 100,000 steps with batch size 64, AdamW, learning rate 10^{-3}, weight decay 0.01, AMP mixed precision, and 1,000 warmup steps. We sweep expected burst lengths $\tau \in \{256, 512, 1024, 2048, 4096\}$ (expected gradient steps per burst in $\{4, 8, 16, 32, 64\}$) using 4-5 seeds per condition.

SA computation. We compute Successor Advantage as in Section B.5, using held-out per-class image-modeling loss. The raw SA compares the successor digit class to the other non-current, non-successor digit classes. We compute the corrected SA for Ring by subtracting the SA of the paired Shuffled model using the same current and successor labels. This correction removes accidental effects from some digit classes being intrinsically easier to model. The measured raw and corrected SA values are similar.

Results. Table Table 4 confirms that MNIST shows a small but clear first-order anticipation effect once bursts are sufficiently long: anticipation emerges between $\tau = 1024$ and $\tau = 2048$, corresponding to roughly 16–32 gradient steps per class burst.

Expected grad. steps until switch	Ring SA	Bursty SA
4	-0.001 ± 0.001	-0.039 ± 0.037
8	-0.001 ± 0.001	-0.039 ± 0.037
16	$+0.000 \pm 0.002$	-0.039 ± 0.038
32	$\mathbf{+0.013 \pm 0.008}$	-0.036 ± 0.034
64	$\mathbf{+0.028 \pm 0.011}$	-0.035 ± 0.037

Table 4: **Anticipation (positive SA) exists for MNIST image modeling.** Values are mean $\pm$ 1 std across 4-5 seeds. For each seed, SA is first averaged over the last 25% of training.

These results should be treated as existence results for anticipation in image modeling. The experimental setup was not optimized extensively. The effect is smaller than in the Wikipedia experiments, but it appears under the same qualitative conditions: predictable first-order transitions, sufficiently long bursts, and per-class held-out evaluation.

D.5. The effect of the optimizer

Theorem 3.1 identified, alongside the anticipation-driving gradient alignment term (Term 2), other implicit regularizers that may modulate the gradient alignment effect. We study how the choice of optimizer effects the emergence of anticipation, through the lens of the implicit regularization interpretation.

We focus on our main experiment configuration, and ablate the optimizer. Table 5 shows the optimizers tested, alongside with suitable learning rates found in small hyperparameter sweeps. For SGD and Adagrad, a weight decay of 0 was used as the default value of 0.1 resulted in significantly worse training losses.

Results. Only AdamW and AdagradW showed anticipation (positive SA) and positive training advantage due to anticipation. Lion and Muon trained well, but did not show anticipation.

Optimizer	LR	Weight decay	SA ($\pm$std)	Training advantage
SGD	5×10^{-5}	0.0	-0.0096 ± 0.0026	-0.0132
SGD	5×10^{-5}	0.1	-0.0010 ± 0.0005	-0.0672
AdamW	5×10^{-5}	0.1	$\mathbf{+0.2730 \pm 0.0194}$	$\mathbf{+0.0903}$
AdamW (no momentum)	3×10^{-4}	0.1	$\mathbf{+0.1846 \pm 0.0170}$	$\mathbf{+0.1771}$
AdagradW	1×10^{-2}	0.0	$\mathbf{+0.2297 \pm 0.0167}$	$\mathbf{+0.1502}$
Lion	1×10^{-5}	0.1	-0.0227 ± 0.0019	-0.0518
Muon	1×10^{-3}	0.1	-0.0104 ± 0.0417	-0.0122

Table 5: **The effect of the optimizer on anticipation.** Only AdamW and AdagradW anticipate (positive SA) and achieve a positive training advantage in consequence. Shuffle-corrected SA is averaged over evaluations in the last 25% of training and reported as mean $\pm$ across-seed standard deviation (12 seeds per setting). Training advantage is based on Theorem 2.2 and is computed as: Shuffled train loss minus Ring train loss, using late-window train-loss means.

Measuring Term 3 interference. To test whether optimizer-dependent anticipation is explained by curvature-driven interference, we directly estimated the alignment term (Term 2) and the order-dependent interference term (Term 3) during SGD and AdamW training, using 12 independent seeds per optimizer. We logged these quantities every 5000 iterations using Hessian-vector products on adjacent class pairs $c_{\text{current}} \to c_{\text{next}}$. For each pair, we computed class gradients and Hessian-vector products with the corresponding class Hessians, giving

$$T_3 = \frac{1}{2} H^{(\text{current})} G^{(\text{next})} - \frac{1}{2} H^{(\text{next})} G^{(\text{current})}.$$

Term 2 was computed using the equivalent expanded form of the alignment-gradient term,

$$T_2 = -\frac{1}{4} H^{(\text{current})} \big(G^{(\text{current})} - G^{(\text{next})}\big) + \frac{1}{4} H^{(\text{next})} \big(G^{(\text{current})} - G^{(\text{next})}\big).$$

For AdamW, we additionally measured the optimizer-effective geometry induced by its diagonal second-moment preconditioner $P = \operatorname{diag}\big((\hat{v} + \epsilon)^{-1/2}\big)$, where $\hat{v}$ is AdamW's bias-corrected second-moment accumulator. We therefore computed

$$T_{3,P} = \frac{1}{2} P H^{(\text{current})} P G^{(\text{next})} - \frac{1}{2} P H^{(\text{next})} P G^{(\text{current})},$$

and $T_{2,P}$ analogously. We report the raw interference ratio $\|T_3\|/\|T_2\|$ and, for AdamW, the optimizer-effective ratio $\|T_{3,P}\|/\|T_{2,P}\|$.

Figure 8 supports the hypothesis that anticipation emerges when Term 3 is small relative to Term 2. The left panel shows that SGD remains in a high-interference regime, with $\|T_3\|/\|T_2\| >> 1$, whereas AdamW training moves the raw ratio close to or below 1. Moreover, AdamW's preconditioner further reduces the optimizer-effective ratio, meaning that the update rule sees an even lower-interference geometry than is visible in raw parameter space. This is further supported by the cosine

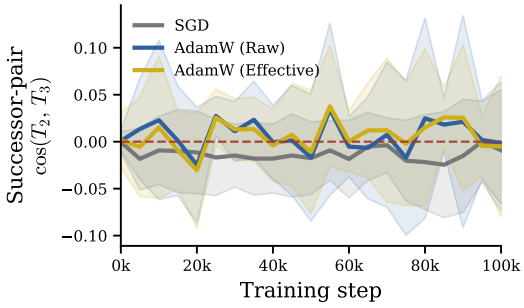

Figure 9: **Cosine similarity evolution of Term 2 and Term 3.**

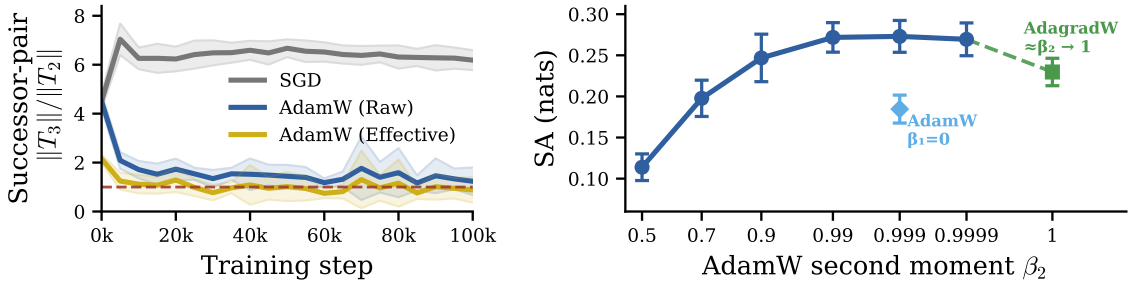

Figure 8: **AdamW enables anticipation through persistent second moment scaling.** Results are averaged over 12 seeds. *(Left) Evolution of the $\|$Term 3$\|/\|$Term 2$\|$ ratio.* AdamW enters a low interference regime, as demonstrated by low raw ratios. Preconditioning further reduces the ratio. *(Right) The key ingredient to anticipation is likely the second moment β_2.* Anticipation generally increases with β_2 and happens even without momentum ($\beta_1 = 0$), suggesting that β_2 is the main driving force.

similarities of the flattened vectors, shown in Figure 9. Thus, AdamW appears to help in two related ways: it reaches parameter regions where raw Term 3 is smaller relative to Term 2, and its adaptive scaling further suppresses the effective interference seen by the optimizer.

The right panel of Figure 8 probes the role of β_2, which controls the timescale of AdamW's second-moment accumulator and hence the stability of the preconditioner. Increasing β_2 increases successor anticipation up to the long-timescale regime around $\beta_2 = 0.999$, consistent with the view that a stable estimate of coordinate-wise gradient scale is important for reducing effective Term 3 interference. AdagradW's cumulative squared-gradient accumulator can be viewed as a limiting long-timescale second-moment accumulator [29], and it also produces strong anticipation. Finally, anticipation persists when $\beta_1 = 0$, showing that first-moment momentum is not necessary for the effect. We therefore interpret β_1 primarily as an optimization aid rather than the source of the anticipation.

Relationship to Yang et al. [33]. It is important to note that Yang et al. [33] observed anticipatory recovery for SGD too, although the effect was stronger for Adam. An explanation of the discrepancy could be that same-domain individual-documents (the domain was CNN news) naturally have more similar Hessians than samples from different languages, and that pretraining and scale further reduces the Hessian mismatch.

D.6. The effect of other hyperparameters

We ablate four hyperparameters of the default configuration reported in Table 2: the number of gradient steps per class (Figure 10), batch size (Figure 12), context length (Figure 11), and model size (Figure 13). We highlight two particularly informative trends below.

Batch size. Figure 12 shows that anticipation increases with batch size, before saturating around a batch size of 64. This trend is consistent with Theorem 3.1: Term 1 scales inversely with batch size and therefore becomes weaker for larger batches. Although Term 1 does not directly shape interactions between classes, the additional gradient noise introduced at small batch sizes appears to oppose anticipation.

Context length. Figure 11 shows that the differences between Ring, Bursty, and Shuffled training diminish as context length increases. In our setup, this is likely because the class becomes identifiable from short prefixes. Once enough tokens have been observed, even non-anticipatory models can infer the current class and predict accordingly, reducing the advantage provided by anticipation.

Importantly, this is not a limitation of anticipation itself, but rather of our class-based construction. If the relevant structure in the training stream were not recoverable from short prefixes, anticipation could remain beneficial at longer context lengths. Such setting is studied by Yang et al. [33], where individual documents rather than broad classes are anticipated. Other similar examples could include anticipation of more document-specific units such as concepts, whose occurrence could become predictable from training order.

We additionally analyze the "forgetting profile" of trained models to study whether adaptation extends beyond the immediate successor class. All ablation figures include forgetting profiles in the right-hand panels. We describe their construction and interpretation below.

Forgetting profiles. The forgetting profile plots show the evaluation loss grouped by relative position in the training stream. For each evaluation, classes are indexed by their distance from the current class, i.e. the class on which the model was most recently trained.

This means that the horizontal axis represents relative, *not* than absolute class identity. In Ring training, this means a rotation of the fixed cyclic order. In Bursty training, it corresponds to the realized order of classes in a particular run. Reported values are averaged over evaluations from the final 25% of training. The axis wraps around cyclically, so positions "-4" and "5" correspond to the same class.

Across ablations, stronger anticipation generally corresponds to larger differences between the Ring and Bursty forgetting profiles. Increasing the number of gradient steps, batch size, or model scale causes both Ring and Bursty models to adapt more strongly to the current class. However, Bursty forgetting profiles are typically asymmetric: classes from the recent past (negative positions) experience less forgetting than distant classes.

The immediate successor class in Bursty training can sometimes exhibit unusually high loss. This is an artifact of the transition structure: Bursty position $+2$-onward can include the "current" class, whereas Ring cannot, as it must first cycle through all other classes.

Ring training exhibits stronger adaptation to the current class, but also substantial adaptation to future classes, including classes beyond the immediate successor. This adaptation decays approximately with future distance in the cycle, producing a more symmetric forgetting profile. This anticipation of future classes comes at the cost of increased forgetting on classes further away in the cycle.

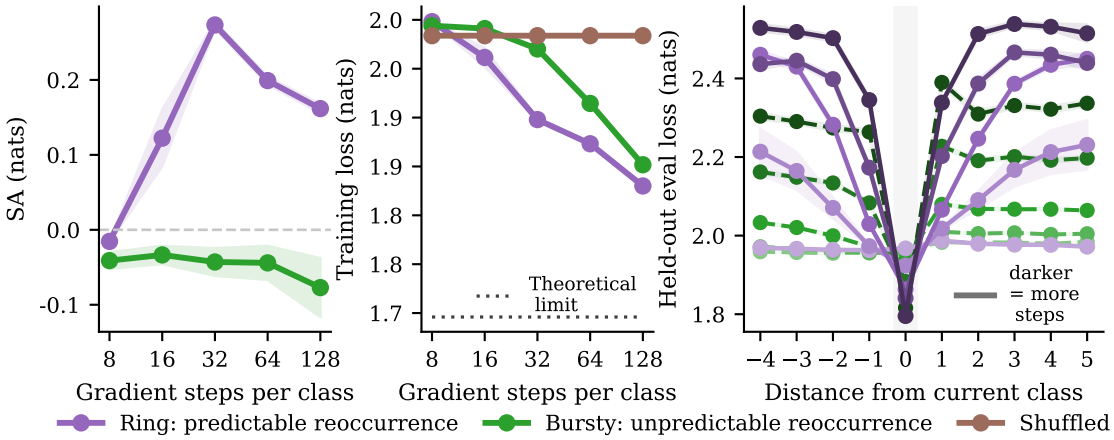

Figure 10: **Effect of the number of gradient steps taken per class.** Reported values are averaged over 12 seeds. *(Left) Effect on Successor Advantage. (Center) Effect on training loss.* Consistent with Yang et al. [33], we observe increasing anticipation up to 32 gradient steps per class. However, increasing the number of steps further is detrimental, likely because tracking the current class becomes the advantageous behavior: Bursty's training loss approaches Ring's. *(Right) Effect on the forgetting profile.* Higher SA means larger adaptation to successor classes, including non-immediate successors in Ring-trained models. Too many gradient steps cause overadaptation to the current class, which comes with a loss of anticipation.

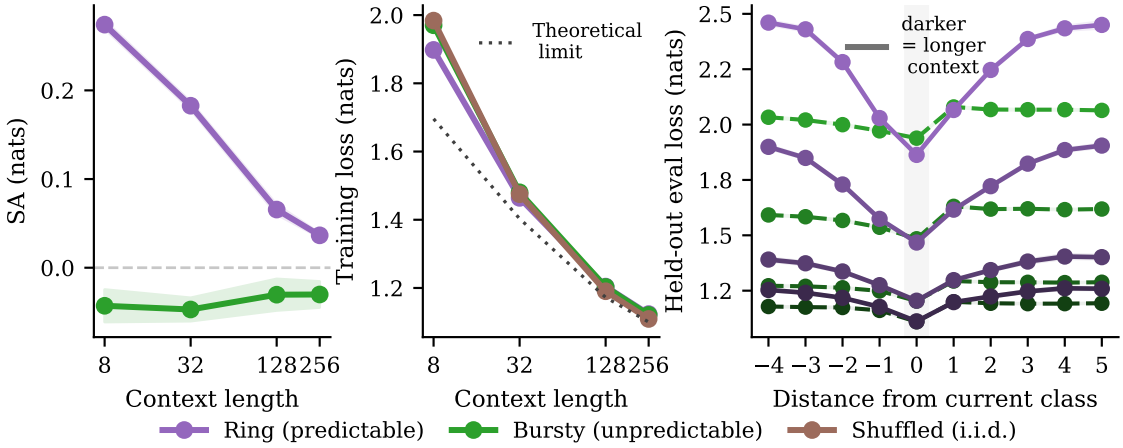

Figure 11: **Effect of the context length.** Reported values are averaged over at least 11 seeds. *(Left) Effect on Successor Advantage. (Center) Effect on training loss.* Contrary to Yang et al. [33], in our class-based setup, longer context lengths hurt anticipation, likely because the class becomes easily identifiable from the prefix, hence any anticipation advantage persists only for a few tokens. *(Right) Effect on the forgetting profile.* Smaller context lengths in Ring-trained models mean larger adaptation, relative to Bursty, to current and successor classes, including non-immediate successors.

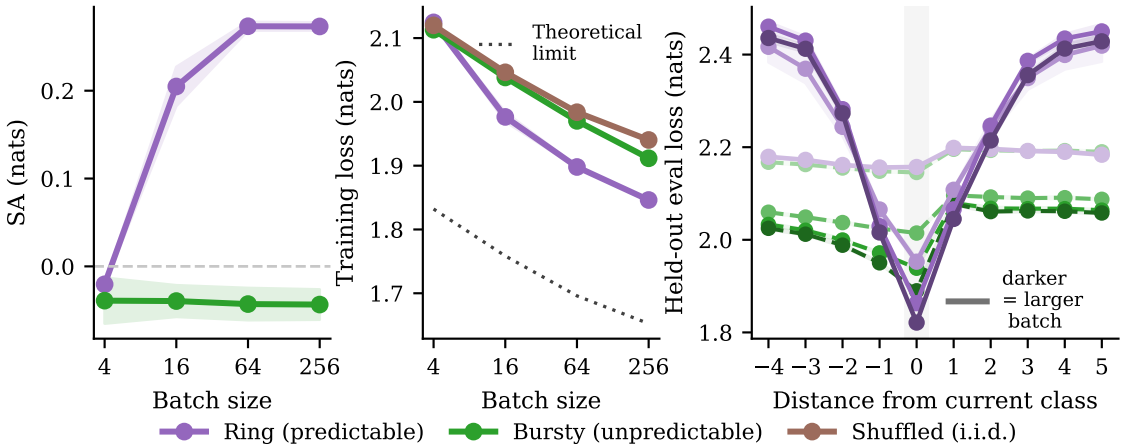

Figure 12: **Effect of the batch size.** Reported values are averaged over 12 seeds. *(Left) Effect on Successor Advantage. (Center) Effect on training loss.* Larger batch size generally helps anticipation, though the difference between batch size 64 and 256 in SA is negligible. *(Right) Effect on the forgetting profile.* Larger batches in Ring-trained models mean larger adaptation, relative to Bursty, to current and successor classes, including non-immediate successors.

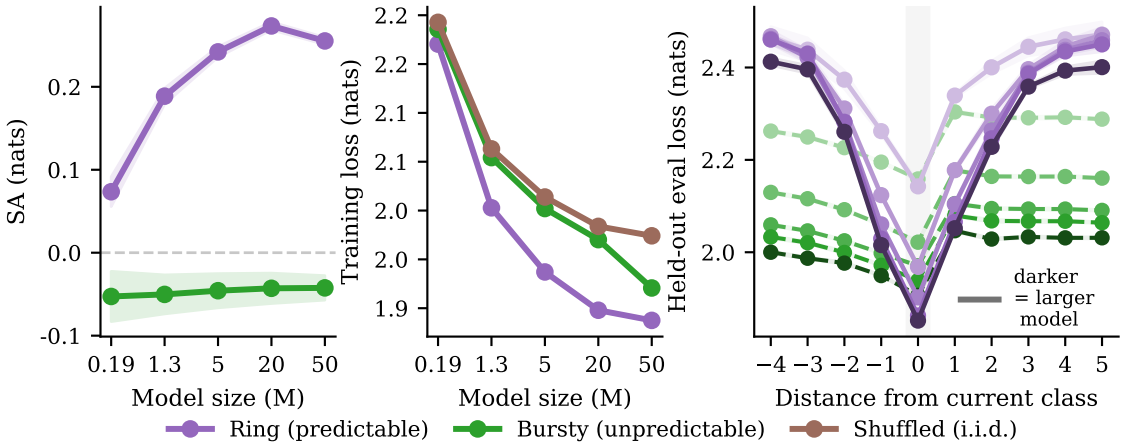

Figure 13: **Effect of the model size.** Reported values are averaged over 12 seeds. *(Left) Effect on Successor Advantage. (Center) Effect on training loss.* Larger model size helps anticipation. *(Right) Effect on the forgetting profile.* Larger Ring-trained models display larger adaptation, relative to Bursty, to current and successor classes, including non-immediate successors.

D.7. Details on the gradient alignment experiments

Measuring gradient alignment. To experimentally verify the SGD implicit bias computed in Theorem 3.1, we estimate the evolution of gradient alignment between consecutive classes $c_{(\text{current})}$ and $c_{(\text{next})}$. Since $-\|G^{(c_{(\text{current})})} - G^{(c_{(\text{next})})}\|^2 = -\|G^{(c_{(\text{current})})}\|^2 + 2G^{(c_{(\text{current})})} \cdot G^{(c_{(\text{next})})} - \|G^{(c_{(\text{next})})}\|^2$,

we measure the cosine similarity defined as

$$Sim(G^{(c_{(\text{current})})}, G^{(c_{(\text{next})})}) = \frac{G^{(c_{(\text{current})})} \cdot G^{(c_{(\text{next})})}}{\|G^{(c_{(\text{current})})}\| \|G^{(c_{(\text{next})})}\|}.$$

This metric removes the confounding effect of different training orders inducing gradients of different magnitudes. We analyze this effect in Figure 18.

We adapt our default experiment, recording gradients at equally spaced intervals. The setup is identical except for two modifications: here the order of classes in the Ring is fixed across seeds, and to see more of the gradient evaluation, we train the model two times longer, for 200k steps. We record gradients for all class pairs (not only consecutive classes).

Measuring the quality of Ring orderings. In order to probe the effect of the specific order of classes chosen, we computed a randomly chosen "Ring: original" ordering, a "Maximum A-priori Similarity" and "Minimum A-priori Similarity" ordering of classes as follows. We created an a-priori influence matrix by recording, for each language pair (i, j), excluding the diagonal, the change in the loss of j, while training on i for 500 steps. The matrix is reported in Figure 14. This estimates the early-influence of training on class i on class j. We centered the rows of the matrix by substracting the row mean. For a given ordering of classes, the sum of influence values corresponding to consecutive classes measures the quality of the ordering, in the sense of how much it respects a-priori class alignments. We then used a Held-Karp dynamic program over subsets to find the minimum- and maximum-weight Hamiltonian cycles in the influence matrix. We call the most optimal ordering "Max Similarity Order" and the most suboptimal "Min Similarity Order". "Max Similarity Order" achieves a total score of -0.312, meaning that training on a class already decreases the loss of the next class, while "Min Similarity Order" achieves $+0.218$, and Ring scores 0.051.

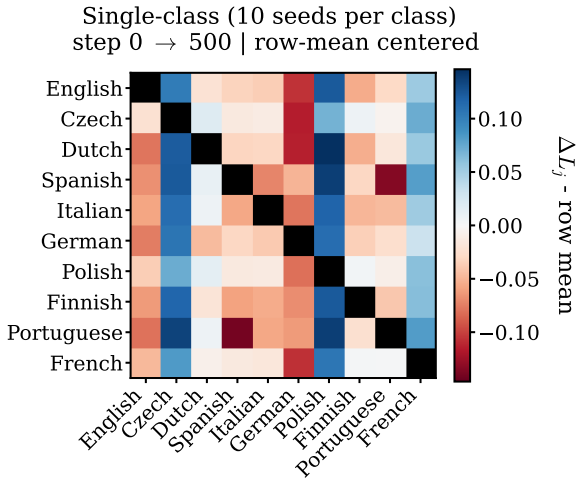

Figure 14: **Matrix of the early-influence of training on class i on class j.** Influences reflect common-sense relationships and origin-based clusters such as Romance languages, Germanic languages, Slavic languages and Uralic languages. Rows are centered via substracting the row mean, values are computed using 10 seeds.

Results. Figure 15 shows the relative (i.e. row-mean-centered) gradient alignments between languages $Sim(G^{(i)}, G^{(j)})$, where $i \neq j$. The plots are not symmetric due to row-mean-centering. Figure 16 shows the evolution of these alignments, i.e. the heatmap recorded after before training has been substracted from the one recorded afer 200k steps.

We observe that for Ring configurations (i.e. "Ring", "Max Similarity" and "Min Similarity"), gradient alignment is concentrated on the off-diagonals (reddest values). The symmetry is due to cosine similarity being symmetric. Bursty and Shuffled generally preserve existing alignments (cosine similarities have low magnitudes), and only reinforce a-priori similarities (e.g. between Czech and Polish).

Figure 17 shows the evolution of gradient similarities between the current class and "next", versus "other" classes (left), and their difference (center). "Max Similarity Order" achieves the highest cosine similarities, but also starts from higher a-priori alignment due to having been optimized for it. The right panel shows that increased gradient alignment correlates with increased SA. Figure 18 shows the same signal, but using $\|g_i - g_j\|^2$, the measurement directly suggested by Theorem 3.1. This experiment also highlights that measuring only the raw $\|g_i - g_j\|^2$ scores is confounded: Ring orders have larger gradient norms, which induces higher raw $\|g_i - g_j\|^2$ scores despite larger cosine similarities (Figure 17). However, measuring the *relative* alignments $\|g_i - g_{\text{next}}\|^2 - \frac{1}{|C|-2}\sum_{j\neq i,\text{next}} \|g_i - g_j\|^2$ reveals the bias towards successor classes too.

The above shows that Ring orders can create alignments between classes that are completely unrelated a-priori, to the extent that a language's gradients may become more aligned to unrelated languages than those most similar to them. An example of this in the "Spanish" row in the "Min Similarity Order" in Figure 15 and Figure 16: Spanish becomes more aligned to Finnish than to Portuguese. *This demonstrates that order can have a significant influence on inter-data relationships between learnt by the model*.

We test the impact of these influences on the training and evaluation losses. As expected, Figure 19 (left) shows "Max Similarity Order" achieves the lowest train loss due to high anticipation. However, this comes with the highest evaluation loss (center and right panels). We note that the fact that Ring orders have high eval loss is not surprising, as they are heavily adapted to the training history, and the evaluation distribution is a mixture, which therefore creates a distribution shift for these models. The fact that "Max Similarity Order" performs even worse than other ordered setups may be due to adapting too much to the next class, to the point that its ability to distinguish the current class from the next class degrades. This is visible on the right panel: even though Spanish and Portuguese are similar languages, due to the training order including the transition Portuguese $\rightarrow$ Spanish, their connection is reinforced to the extent that the model struggles to distinguish them.

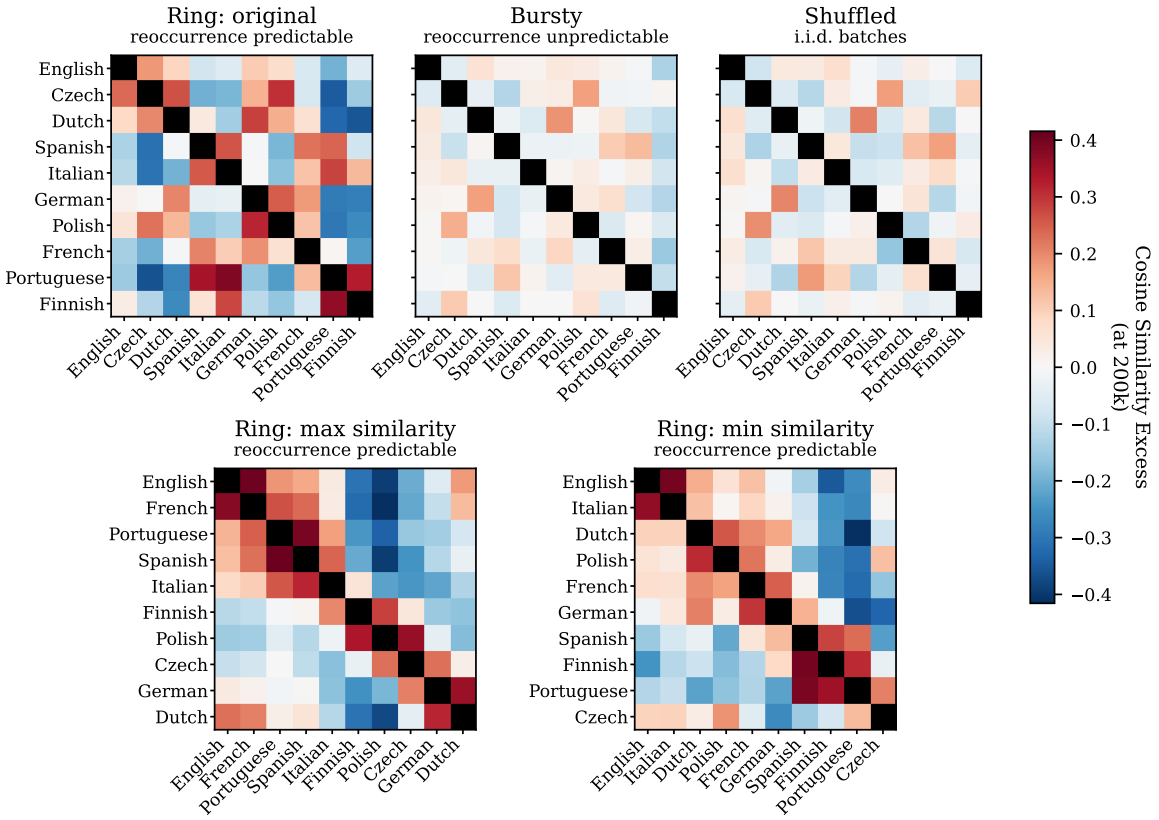

Figure 15: **Heatmaps of the relative gradient cosine similarities at the end of training (200k).** We plot, for each order configuration, the relative gradient similarities between classes after training step 200k. Relative gradient similarities are computed for each (i, j) cell by computing the cosine similarity between G_i and G_j and substracting the mean of the row, not including the cell (i, i). Values are averaged over 12 seeds. The figure shows that Ring orders (Ring, Max Similarity and Min Similarity) create gradient alignment between consecutive classes, even in the case of very different languages like Spanish and Finnish (e.g. the "Spanish" row on the Min Similarity panel).

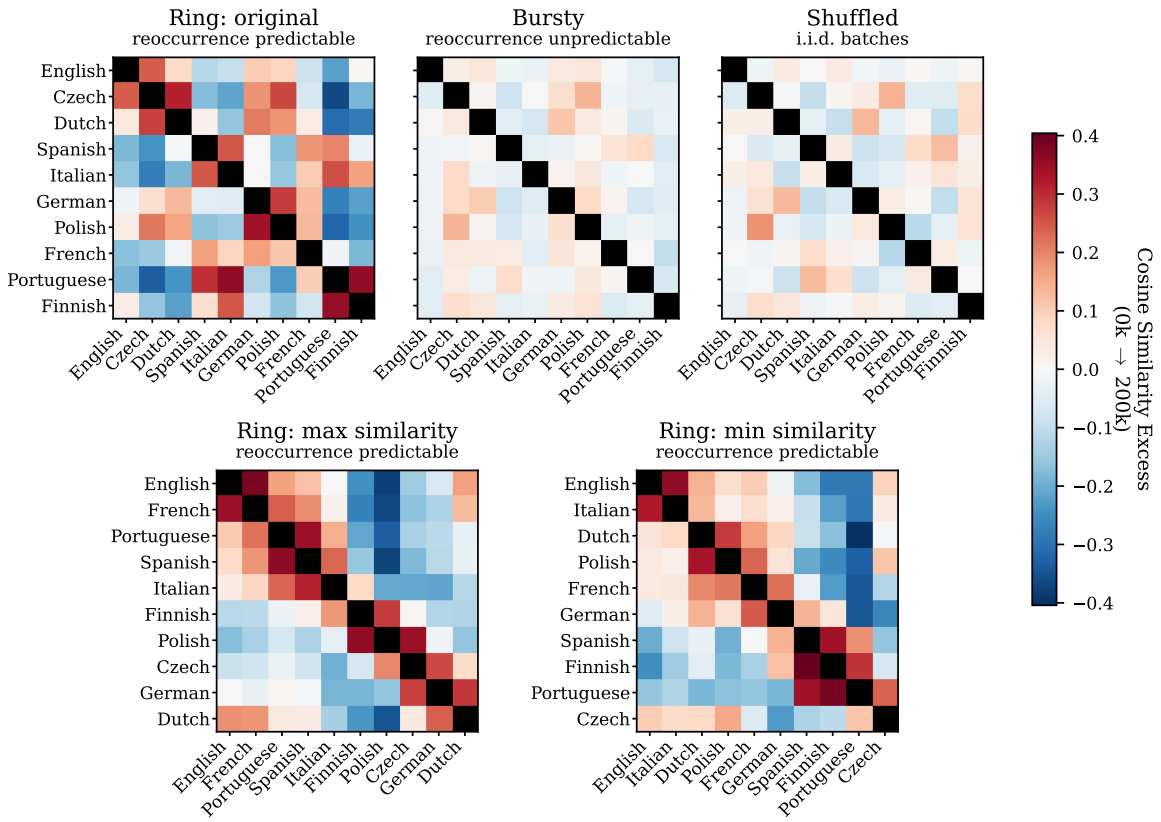

Figure 16: **Heatmaps of the evolution of relative gradient cosine similarities.** We plot, for each order configuration, the difference of relative gradient similarities between training step 200k and step 0. Relative gradient similarities are computed for each (i, j) cell by computing the cosine similarity between G_i and G_j and substracting the mean of the row, not including the cell (i, i). Values are averaged over 12 seeds. The figure shows that Ring orders (Ring, Max Similarity and Min Similarity) create gradient alignment between consecutive classes, even in the case of very different languages like Spanish and Finnish (e.g. the "Spanish" row on the Min Similarity panel).

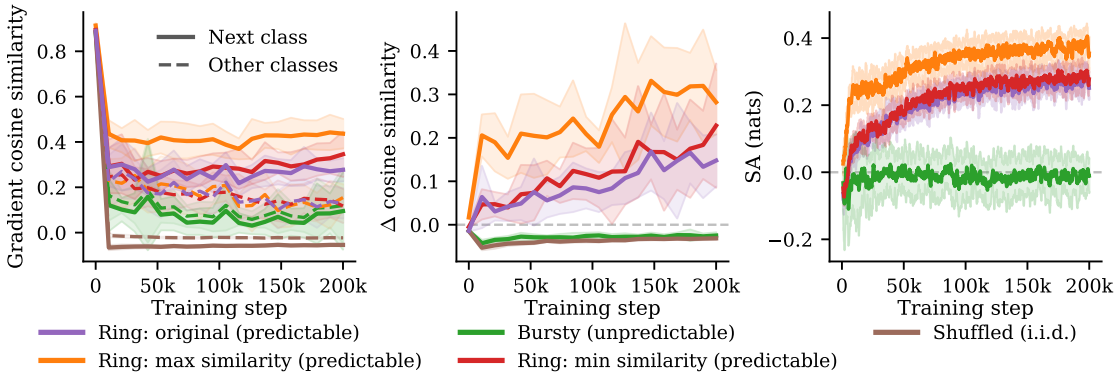

Figure 17: **Evolution of gradient alignment towards successor classes and Anticipation.** Standard deviations are computed from 12 seeds. *(Left) Gradient cosine similarity between the current and next class, and current and other classes and (Center) their difference.* Max Similarity Order achieves largest gradient alignment, while Ring and Min Similarity Order are comparable. *(Right) Successor Advantage during training.* Max Similarity Order achieves highest anticipation.

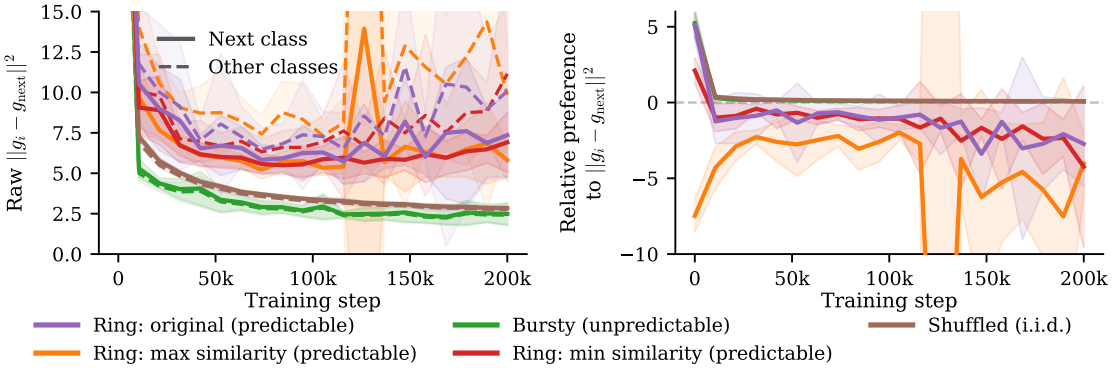

Figure 18: **Evolution of the norm of the gradient difference towards successor classes.** Standard deviations are computed from 12 seeds. *(Left) Norm of the gradient difference between the current and next class, and current and other classes and (Right) their difference.* While the raw $||g_i - g_{\text{next}}||^2$ is larger for predictable orders, plotting the magnitude relative to other, non-successor classes reveals that Ring training aligns successor classes according to this Theorem 3.1-based metric too.

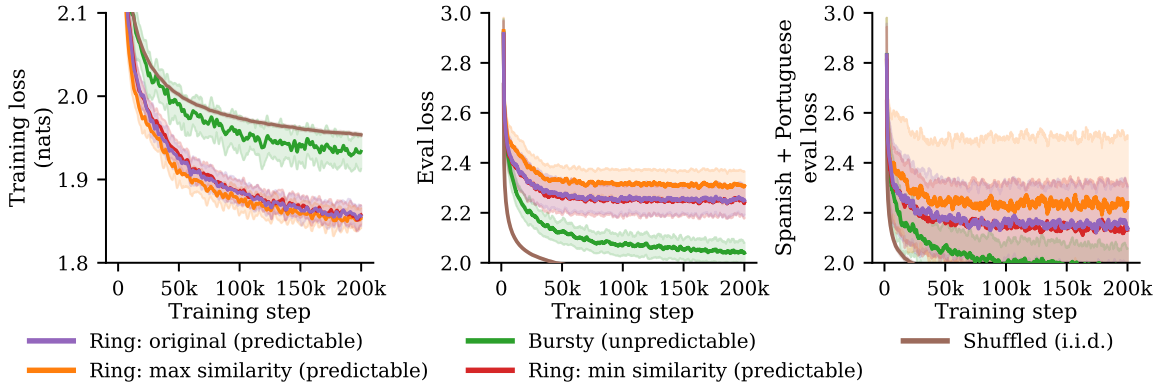

Figure 19: **The effect or class orderings.** Values are averaged over 12 seeds, losses were smoothed with a 20-point moving average. *(Left) Effect on the training loss.* Max Similarity Order has lowest loss. *(Center) Effect on eval loss.* Non-random orderings achieve higher eval loss due to the distribution shift of training on a structured order, but evaluated on a class mixture. Max Similarity Order achieves high loss even among ordered configurations. *(Right) Effect on the eval loss of Spanish and Portuguese documents only.* Max Similarity Order still performs worst, likely because too much consecutive class alignment blurs the ability to distinguish these classes. In the Max Similarity Order, Spanish directly follows Portuguese.