Curriculum Learning as Transport: Training Along Wasserstein Geodesics

Changho Shin[*]
University of Wisconsin–Madison
`cshin23@wisc.edu`

David Alvarez-Melis
Microsoft Research & Harvard University
`daalvare@microsoft.com`

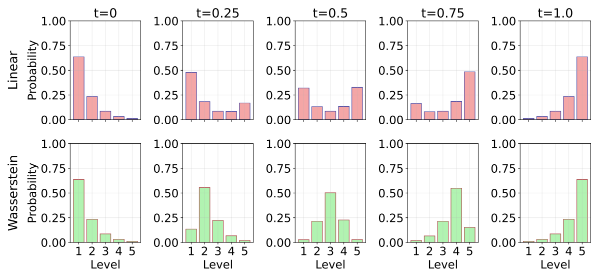
Abstract

Curriculum learning is widely used but remains poorly understood: its success often depends on ad-hoc design choices, making it unclear when and why it helps. We investigate these questions through the lens of Wasserstein curriculum learning—a framework that interpolates between easy and hard data distributions along the Wasserstein geodesic. While prior work has applied this idea in narrow reinforcement learning settings, we present the first controlled study across synthetic tasks and reasoning benchmarks with large language models. Our experiments map how endpoint distributions and traversal rates shape learning, and identify mechanisms by which curricula accelerate progress, especially on difficult examples. Empirically, Wasserstein curricula can match target accuracy with fewer examples than uniform sampling or linear blends, but only under specific conditions. Together, these findings offer new insights into when Wasserstein curricula are effective and why, advancing our understanding of curriculum learning beyond task-specific heuristics.

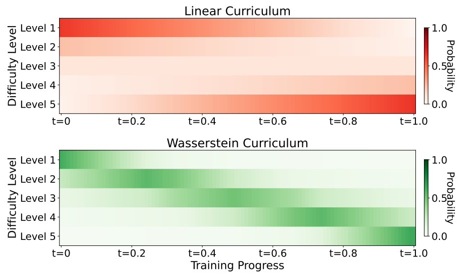
1 Introduction

Curriculum learning aims to build skills progressively by training on easier examples before harder ones, in contrast to traditional, *static* sampling [1]. Despite its intuitive appeal, the effectiveness of curricula often hinges on ad-hoc choices of difficulty measures and pacing rules, leaving open the question of when and why they actually help. A variety of strategies have been proposed—from self-paced selection to teacher–student and bandit schedulers [2, 3, 4]—but these methods remain sensitive to noise and lack principled defaults. In this work, we study curriculum learning through the lens of Wasserstein interpolation. By viewing a curriculum as a trajectory in the space of distributions over difficulty levels, the Wasserstein geodesic provides a natural, smooth way of transitioning between easy and hard data [5, 6], in contrast, for example, to naive linear interpolation. Prior work has demonstrated empirical success of Wasserstein curricula in reinforcement learning settings for specific tasks [7, 8], but their behavior across domains and models remains largely unexplored, and we still lack a mechanistic understanding of how geodesic paths influence learning dynamics. In this work, we seek to characterize conditions under which Wasserstein curricula succeed, and the mechanisms by which they do so. Through controlled experiments on synthetic tasks and reasoning benchmarks with large language models, we map how endpoint distributions and traversal rates shape outcomes, and identify cases where Wasserstein curricula accelerate learning—particularly on challenging examples. Building on these insights, we introduce a curvature-based pacing rule that adapts distribution geometry to regions where models struggle. Together, our findings move beyond task-specific heuristics and point toward principles for more reliable curriculum design.

[*]Work done during a research internship at Microsoft Research.



(a) Linear vs. Wasserstein interpolation.



(b) Curriculum heatmap.

Figure 1: Comparison of linear and Wasserstein curricula. (a) Wasserstein interpolation shifts probability mass smoothly between adjacent levels, producing a natural progression of difficulty, whereas linear interpolation jumps directly between endpoints. (b) Viewing p^t as the batch-level sampling distribution over time shows that linear curricula largely skip intermediate levels, whereas Wasserstein curricula traverse them gradually.

2 Wasserstein–Geodesic Curriculum

We formalize a curriculum as a sequence of training data distributions $\{P^k\}_{k=1}^T$ over difficulty levels $\{\mathcal{D}_1, \dots, \mathcal{D}_m\}$, where each batch is drawn from P^k. A curriculum is then a path $\{P_t\}_{t\in[0,1]}$ between two endpoint distributions $P_0, P_1 \in \Delta^{m-1}$, with $P^k = P_{t_k}, \quad t_k \in [0,1]$.

Wasserstein interpolation. We construct the path using the Wasserstein geodesic: for $t \in [0,1]$,

$$P_t = \arg\min_{Q\in\Delta^{m-1}} \Big\{(1-t)\, W(Q, P_0) + t\, W(Q, P_1)\Big\},$$

which in one dimension reduces to linear interpolation of quantiles [6]. This ensures probability mass *flows* smoothly between adjacent levels rather than jumping abruptly. As illustrated in Figure 1, linear mixing skips over intermediate levels, while the Wasserstein path traverses them in sequence, producing a more natural progression.

Curriculum with Wasserstein interpolation. By setting $P^k = P_{t_k}$, we obtain curricula that are monotone and locality-preserving. The choice of endpoints (P_0, P_1) determines which regions of the difficulty spectrum the curriculum connects, and the schedule $\{t_k\}$ controls how quickly training moves along the path. This provides a simple but flexible framework for designing curricula, with Figure 1 capturing the core intuition.

3 Experiments

Common Setup. We assume a dataset where each example is labeled with a difficulty level $\mathcal{D}_\ell$, for $\ell = 1, \dots, L$. These labels may come from ground truth or be imputed. At training step t, mini-batches are drawn from a distribution $P^t \in \Delta^{L-1}$ over levels. We set $P^t = P_{t/T}$, where $\{P_t\}_{t\in[0,1]}$ is the Wasserstein interpolation between two endpoints $P_0, P_1 \in \Delta^{L-1}$. The endpoints are defined by an *exponentially increasing distribution* over levels: $(P_1)_\ell = \frac{\exp(\ell/\tau)}{\sum_{j=1}^{L}\exp(j/\tau)}, (P_0)_\ell = (P_1)_{L-\ell+1}$. The temperature τ controls sharpness: low τ concentrates probability on higher levels, while high τ yields a flatter distribution. P_0 is the symmetric counterpart of P_1, emphasizing easier levels instead of harder ones. We compare against two baselines: (i) *uniform* sampling over levels, and (ii) a *linear interpolation* $P^t_{\text{lin}} = (1 - t/T)\, P_0 + (t/T)\, P_1$ using the same endpoints.

Tasks Construction. We describe tasks and difficulty level construction briefly here. For further details, refer to Appendix A.

k**-Parity. (Appendix A.1)** Input $x \in \{0,1\}^d$, label $y = \bigoplus_{i\in S} x_i$ for hidden $|S| = k$. Level ℓ uses a nested subset S_ℓ with $|S_\ell| = \ell$ and $S_1 \subset \cdots \subset S_k$.
Dyck (bracket completion). (Appendix A.2) Complete bracket strings to be balanced; level ℓ increases total length and the number of blanks to fill (thus higher nesting).
Survo. (A.3) Fill an $n\times n$ grid to match row/column sums; level ℓ increases empty cells.

Models. We use a 2-layer MLP for k-Parity and 3- and 6-layer Transformers for Dyck and Survo.

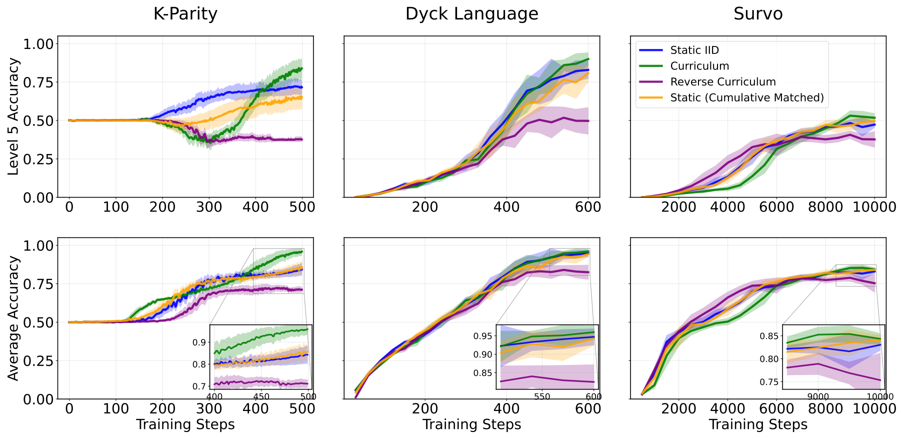


Figure 2: Curriculum vs. static i.i.d., reverse (hard→easy), and static-matched baselines. Crriculum improves over i.i.d., while the reverse underperforms—showing that *ordering* matters. static-matched also often trails behind, indicating that gains are not due to total exposure alone.

3.1 How Much Does the Curriculum Choice Matter?

Can the choice of curriculum schedule change training outcomes? We show that curricula can both help and harm, highlighting the importance of ordering and training dynamics.

Setup. We compare four schedules: (i) static i.i.d., (ii) Wasserstein, (iii) reverse (hard→easy), and (iv) static-matched, which fixes per-level probabilities to match the Wasserstein endpoint distribution.

Findings. Figure 2 shows the Wasserstein curriculum outperforms i.i.d. sampling, while the reverse schedule underperforms—highlighting the importance of curriculum *ordering*. The static-matched baseline also lags behind, indicating that benefits come from the *path* of progression rather than final cumulative exposure alone (see Appendix B.1).

3.2 Comparison with Linear Interpolation

Setup. We compare the Wasserstein curriculum against a linear interpolation baseline.

Findings. Figure 3 shows that Wasserstein traverses intermediate levels smoothly, which benefits k-Parity by improving performance at harder levels. In contrast, Dyck and Survo show little difference between the two curricula. This suggests that the effect of skipping or covering intermediate levels may vary with the structure of the task.

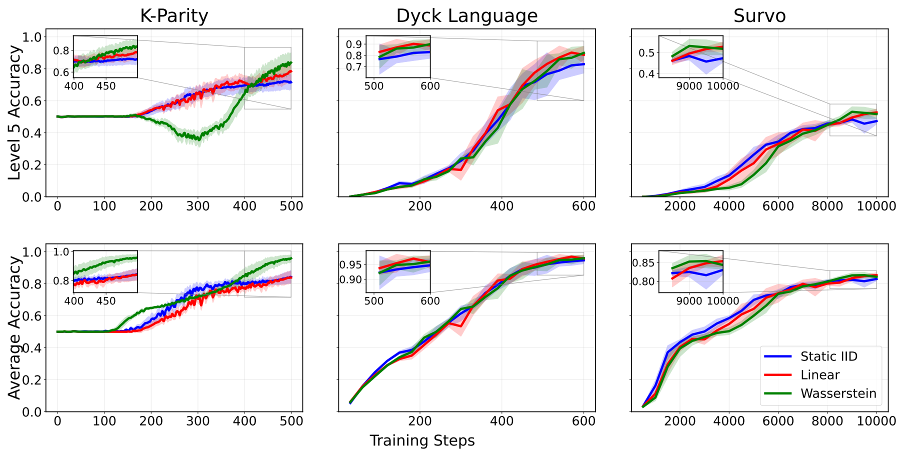
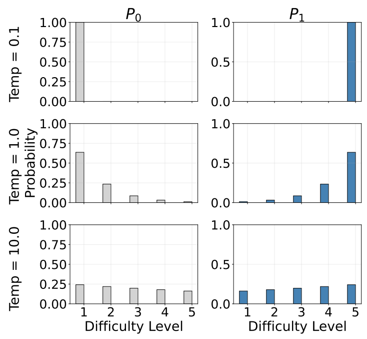
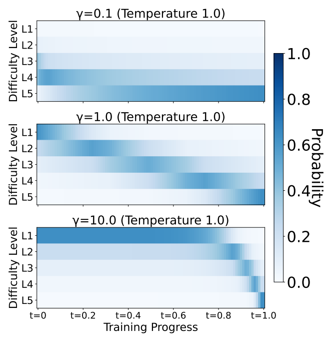
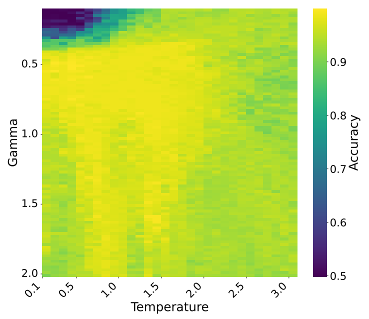


Figure 3: Static i.i.d., linear, and Wasserstein curricula. Wasserstein improves over baselines on k-Parity, while on Dyck and Survo it is comparable to linear interpolation, indicating that the effect varies across tasks.







(a) Endpoints vs. temperature τ. (b) Batch-level path for γ. (c) Accuracy across (τ, γ).

Figure 4: **(a)** Effect of τ: lower values make endpoints P_0, P_1 more peaked, higher values flatten them. **(b)** Effect of γ: batch-level trajectories under $s = t^\gamma$. **(c)** Accuracy across (τ, γ): performance collapses for low τ, small γ (fast shift to P_1); elsewhere accuracy changes smoothly.

3.3 What Factors Govern Curriculum Effectiveness?

How do endpoint shape and traversal rate affect performance? We investigate how the *shape* of the endpoints and the *speed* of traversal along the path affect outcomes.

Setup. We vary τ to set endpoint flatness and γ to set traversal speed via $t' = t^\gamma$ in k-Parity task.

Findings. Figure 4(c) shows that low τ and small γ (top left) lead to catastrophic performance collapse by shifting too quickly to P_1. Moderate values yield a broad "sweet spot" where exposure is balanced and accuracy improves, with τ=1, γ=1 a robust default.

3.4 Designing Pacing Geometry

The τ, γ study (Section 3.3) shows that shifting too quickly toward P_1 (low τ, small γ) can cause collapse before prerequisite levels are mastered, so pacing must be controlled. We treat pacing as the geometry of level positions along the Wasserstein path: uniformly spaced levels yield near-linear progression, while left-heavy layouts prolong exposure to easier levels and right-heavy layouts accelerate progression. Using diagnostics from uniform training, we reshape this geometry by allocating more exposure before bottlenecks (Figure 5) via the Curvature-Based Pacing procedure in Algorithm 1.

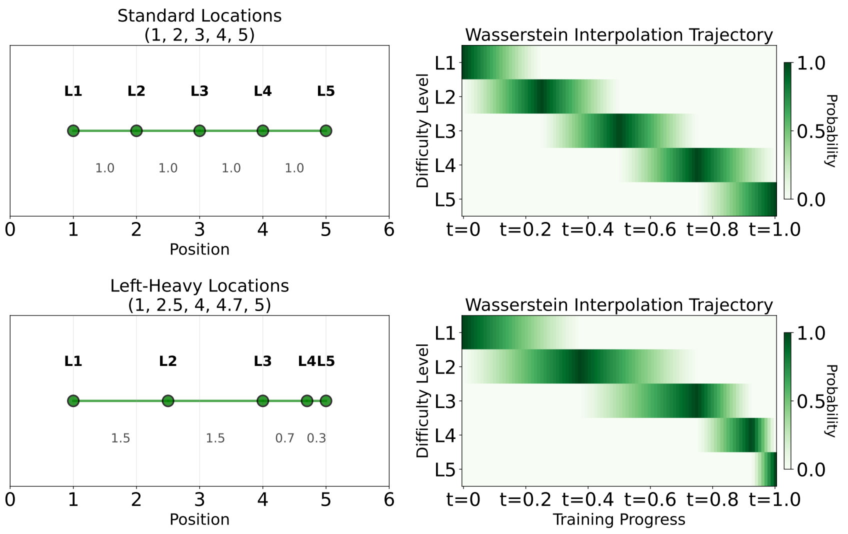


Figure 5: **Effect of geometry on Wasserstein curricula.** Uniform spacing (top) yields near-linear pacing; left-heavy spacing (bottom) prolongs exposure to early levels and reduces collapse risk.

Setup. Let m_ℓ be the accuracy at level ℓ under static i.i.d. sampling, and define $d_\ell = 1 - m_\ell$. Curvature flags stalls: $\kappa_\ell = \max\{0,\, d_{\ell+1} - 2d_\ell + d_{\ell-1}\}, \kappa_1 = \kappa_L = 0$. We convert curvature to pacing weights, $\pi_\ell \propto (1-\lambda)\kappa_\ell + \lambda/L, \quad \sum_{\ell=1}^{L} \pi_\ell = 1$, then allocate $N_\ell \approx B\pi_\ell$ and map cumulative proportions to schedule points $t_{1:L} \in [0,1]$. This *Curvature-Based Pacing (CBP)* reshapes the level geometry from training dynamics and is detailed in Algorithm 1.

Algorithm 1 Curvature-Based Pacing (CBP)

```
Require: Accuracies $m_{1:L}$, budget $B$, smoothing $\lambda$, anchor $a$=1, span $S$=$L$−1
Ensure: Schedule $t_{1:L}$, counts $N_{1:L}$, exposures $\pi_{1:L}$
1: $d_\ell \leftarrow 1 - m_\ell$;   $\kappa_\ell \leftarrow \max\{0,\, d_{\ell+1} - 2d_\ell + d_{\ell-1}\}$ with $\kappa_1$=$\kappa_L$=0
2: $\pi_\ell \leftarrow (1-\lambda)\kappa_\ell + \lambda/L$;   normalize $\sum_\ell \pi_\ell = 1$
3: $N_\ell \leftarrow \mathrm{round}(B\,\pi_\ell)$
4: $g_1 \leftarrow 2\pi_1$;   for $\ell$=2:$L$−1: $g_\ell \leftarrow 2\pi_\ell - g_{\ell-1}$
5: $G \leftarrow \sum_{\ell=1}^{L-1} g_\ell$;   $g_\ell \leftarrow g_\ell \cdot \frac{S}{G}$
6: $t_1 \leftarrow a$;   for $\ell$=1:$L$−1: $t_{\ell+1} \leftarrow t_\ell + g_\ell$
7: return $(t_{1:L},\, N_{1:L},\, \pi_{1:L})$
```

Findings. Figure 6 presents results in the small-data regime where the training budget is limited. Adaptive pacing (CBP) is generally beneficial under such constraints, particularly by improving accuracy at intermediate levels.

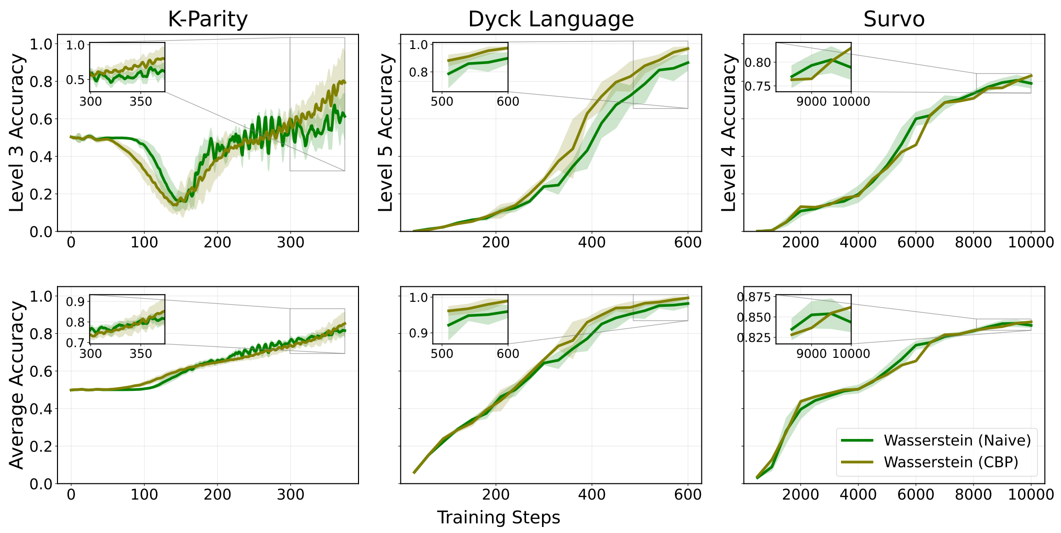


Figure 6: Comparison of curvature-based pacing methods in the small-data regime. Results highlight improvements in intermediate-level accuracy when training budgets are limited.

4 Related Work

Curriculum learning and difficulty-aware training. Classical curriculum learning moves from easy→hard and can stabilize optimization and improve generalization [1]. Automation has focused on what to show and when: self-paced learning matches example difficulty to learner competence [2]; teacher–student curricula prioritize items with high learning progress [3]; bandit-style schedulers adapt task proportions online [4]; and competence-based schedules modulate how quickly harder content is introduced [9]. Evidence suggests pacing strongly affects outcomes [10], while training-dynamics diagnostics provide signals for difficulty and ambiguity [11]; related heuristics include hard-example mining and forgetting-based selection [12, 13]. In contrast, WARP (i) models difficulty at the *distribution* level over discrete difficulty levels and defines the curriculum as a path in this space, and (ii) decouples the *path* (fixed, principled) from *pacing* (adaptive, via lightweight progress signals). This separation yields smooth, locality-preserving transitions while retaining simple, robust control over when to advance.

Recent evidence also probes *when* curricula help. [14] show that gains often depend on limited training budgets and label noise, with random ordering sometimes matching or exceeding curated curricula, and anti-curricula generally harmful. More recently, [15] analyze data-ordering effects in LLM mathematical reasoning, comparing forward and reverse curricula across several difficulty signals (problem difficulty, surprisal, margin, and uncertainty), and find that the optimal ordering varies with model capability and task complexity. Our experimental results largely align with these observations: Wasserstein curricula are most effective under limited budgets, reverse ordering degrades performance, and pacing dynamics critically influence learning outcomes.

Optimal transport. Optimal transport measures distances between probability distributions via the minimal cost of moving mass; in this metric, Wasserstein (displacement) interpolation follows geodesics connecting two endpoint distributions [5, 6, 16]. OT has also been used to *construct curricula* in reinforcement learning: by generating geodesic sequences or constrained OT plans between task distributions [7, 8]. Our approach is related in that it also uses Wasserstein geodesics for curriculum learning, but we extend the scope beyond reinforcement learning to a general framework that applies to pre-training, post-training, and beyond. Moreover, we investigate the broader design space of curricula defined as Wasserstein geodesics, analyzing how endpoint choice and traversal speed shape training outcomes.

Data Mixing. Designing training distributions from heterogeneous sources is central to large-scale model training. Static approaches fix mixture weights in advance, e.g., DoReMi [17] optimizes domain sampling via a proxy model, and DoGE [18] estimates domain importance from small runs. Adaptive approaches adjust weights online, using optimization signals or reinforcement learning [19, 20]. A recent direction predicts mixture outcomes: REGMIX regresses performance from proxy runs [21], data mixing laws model weight–loss functions [22], and mixtures of data experts approximate source contributions [23]. Other work goes beyond source-level mixing, such as topic-based mixtures [24] or regroup-and-balance strategies [25]. Our work shares the perspective of viewing curricula as data mixing, but differs in two ways: (i) we emphasize *distributional paths* rather than static or adaptive reweighting, and (ii) we study principled design factors such as endpoint choice and pacing, complementing existing online weighting methods.

5 Conclusion

We studied curricula defined by Wasserstein interpolation between endpoint difficulty distributions. Across tasks, this produces smooth progressions whose effectiveness varies with endpoint design and pacing. Our analysis shows that ordering and training dynamics—not just total exposure—shape outcomes, clarifying when Wasserstein curricula help and when simpler baselines suffice.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[2] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, 2010.

[3] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. In *arXiv preprint arXiv:1707.00183*, 2017.

[4] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *ICML*, pages 1311–1320, 2017.

[5] Robert J. McCann. A convexity principle for interacting gases. *Advances in Mathematics*, 128(1):153–179, 1997.

[6] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[7] Peide Huang, Mengdi Xu, Jiacheng Zhu, Laixi Shi, Fei Fang, and Ding Zhao. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. In *Advances in Neural Information Processing Systems*, 2022. NeurIPS.

[8] Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11341–11358. PMLR, 2022.

[9] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Mihai Păsărescu, and Tom Mitchell. Competence-based curriculum learning for neural machine translation. In *NAACL*, pages 1162–1172, 2019.

[10] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, pages 2535–2544, 2019.

[11] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *EMNLP*, pages 9275–9293, 2020.

[12] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016.

[13] Mariya Toneva, Alessandro Sordoni, Pietro Perona, Vithursan Birodkar, Zaid Harchaoui, and Yejin Choi*? Koyama. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

[14] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *International Conference on Learning Representations*, 2021.

[15] Yaning Jia, Chunhui Zhang, Xingjian Diao, Xiangchi Yuan, Zhongyu Ouyang, et al. What makes a good curriculum? disentangling the effects of data ordering on llm mathematical reasoning. *arXiv preprint arXiv:2510.19099*, 2025.

[16] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Birkhäuser Cham, 2015.

[17] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*, 2023.

[18] Simin Fan, Matteo Pagliardini, and Martin Jaggi. DoGE: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393*, 2023.

[19] Abhishek Kumar et al. Mixture proportion estimation via importance weighting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Tongzhou Wu et al. Skill: Adaptive data mixtures for pretraining. In *International Conference on Learning Representations (ICLR) Workshop*, 2023.

[21] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. REGMIX: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.

[22] Jiasheng Ye, Samira Abnar, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024.

[23] Lior Belenki, Alekh Agarwal, Tianze Shi, and Kristina Toutanova. Optimizing pre-training data mixtures with mixtures of data expert models. In *Proceedings of the 63rd Annual Meeting of the ACL (Long Papers)*, 2025.

[24] Jiahui Peng, Xinlin Zhuang, Jiantao Qiu, Ren Ma, Jing Yu, He Zhu, and Conghui He. Topic over source: The key to effective data mixing for language model pre-training. *arXiv preprint arXiv:2502.16802*, 2025.

[25] Albert Ge, Tzu-Heng Huang, John Cooper, Avi Trost, Ziyi Chu, Satya S. Namburi, Ziyang Cai, Kendall Park, Nicholas Roberts, and Frederic Sala. R&b: Domain regrouping and data mixture balancing for efficient foundation model training. *arXiv preprint arXiv:2505.00358*, 2025.

[26] Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, et al. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *arXiv preprint arXiv:2505.19641*, 2025.

A Experiment Setup Details

We include detailed experiment setups and data examples.

A.1 k-Parity Dataset

Task. Each example is a binary string $x \in \{0,1\}^n$ with label given by the parity of the first k bits:

$$y = f_k(x) = \bigoplus_{i=1}^{k} x_i,$$

where $\oplus$ denotes XOR. The remaining bits act as nuisance features and are independent of the label. We report classification accuracy at each level.

Construction. We fix n=32 and define five nested distributions $\mathcal{S}_1 \subset \cdots \subset \mathcal{S}_5$ over inputs. At level $k \leq 4$, the first k bits are i.i.d. $\mathrm{Bernoulli}(1/2)$, the next $(5-k)$ bits are fixed to 0, and the remaining $n-5$ tail bits are i.i.d. $\mathrm{Bernoulli}(1/2)$. At level 5, all n bits are i.i.d. $\mathrm{Bernoulli}(1/2)$. Training examples at level k are drawn uniformly from $\mathcal{S}_k$ and labeled by $f_k(x)$.

Difficulty levels. Levels correspond to the number of informative prefix bits:

level 1: k=1, level 2: k=2, level 3: k=3, level 4: k=4, level 5: k=5.

Evaluation slices. Because the supports are nested, we evaluate each level on its *unique slice*

$$\mathcal{U}_k = \mathcal{S}_k \setminus \mathcal{S}_{k-1} \qquad (\mathcal{S}_0 := \varnothing),$$

so that accuracy reflects performance on examples newly introduced at level k. Concretely, for $k \leq 5$,

$$\mathcal{U}_k = \{x : \ x_k = 1, \ x_{k+1} = \cdots = x_5 = 0\},$$

with all remaining bits drawn i.i.d. $\mathrm{Bernoulli}(1/2)$.

Worked example (level 3 unique slice). We illustrate with n=8 for clarity (the actual setup uses n=32). On $\mathcal{U}_3$, we must have x_3=1 and x_4=x_5=0; all other bits are unconstrained. Use colors to mark roles: blue = previously informative bits, orange = the new level-3 bit, gray = constrained zeros, black = nuisance tail.

Input example:

$$x = \big[\ \texttt{1 0 1 0 0 1 0 1}\ \big].$$

Then

$$f_3(x) = x_1 \oplus x_2 \oplus x_3 = 1 \oplus 0 \oplus 1 = 0, \qquad f_2(x) = x_1 \oplus x_2 = 1 \oplus 0 = 1.$$

A predictor that has only learned f_2 (from level 2) will *systematically* flip the label on $\mathcal{U}_3$.

Lemma (systematic error on unique slices). For $k \in \{2,3,4,5\}$, the predictor $\hat{y}(x) = f_{k-1}(x)$ achieves 0% accuracy on $\mathcal{U}_k$.

Proof. On $\mathcal{U}_k$ we have $x_k = 1$ and $x_{k+1} = \cdots = x_5 = 0$. Thus $f_k(x) = \big(\bigoplus_{i=1}^{k-1} x_i\big) \oplus x_k = f_{k-1}(x) \oplus 1$, so $\hat{y}$ is flipped on every example. □

Dataset size. We evaluate three training budgets: 375 steps (small), 500 steps (medium), and 1500 steps (large), using a batch size of 1000. Results in the main text are reported for the medium budget, with additional regimes in Appendix B.

A.2 Dyck Language Dataset

Task. Each example is a *prefix-completion* problem over the Dyck language (well-nested brackets). Let the hidden solution be a valid bracket string $w^\star$ of total length L over an alphabet of $n_{\text{types}} = 4$ bracket pairs. We reveal the prefix $x = w^\star_{1:L-m}$ and ask the model to predict the *exact* length-m suffix $y = s^\star = w^\star_{L-m+1:L}$ such that xy is a valid Dyck word. We report *exact-match* accuracy on the completion y.

Construction. We follow SynLogic [26][2]. The generator is parameterized by: (i) the target total length L, and (ii) the suffix length m to be filled. Sequences draw from a multi-type bracket alphabet that includes (not limited to) $()$, $[]$, $\{\}$, and $\langle\rangle$. For each level, we sample $w^\star$, expose its prefix of length $L-m$ as the input, and use the final m tokens as the label. (The suffix can contain a mix of opens and closes; it is always the *contiguous* tail segment of length m.)

Difficulty levels. We vary (L, m) as:

level 1: $(L{=}15,\ m{=}3)$, level 2: $(L{=}18,\ m{=}4)$, level 3: $(L{=}21,\ m{=}5)$,
level 4: $(L{=}24,\ m{=}6)$, level 5: $(L{=}27,\ m{=}7)$.

Examples (colored completions). Blue = input prefix; orange = missing suffix (only closers, hence unique). Requires `\usepackage{xcolor}`.

- **Level 1**:
  Input `<[](){<([])` *Completion* `>}>`
  Full target: `<[](){<([])>}>`
- **Level 2**:
  Input `{(«><[([[]])]` *Completion* `»)}`
  Full target: `{(«><[([[]])] »)}`
- **Level 3**:
  Input `{{{[[(())]]([[]` *Completion* `])}}}`
  Full target: `{{{[[(())]]([[]])}}}`
- **Level 4**:
  Input `(({()<>[][«{([])}` *Completion* `»]}))`
  Full target: `(({()<>[][«{([])}»]}))`
- **Level 5** (fixed to be well-formed):
  Input `(<>({})[]{(){{[{[([]` *Completion* `)]}]}}})`
  Full target: `(<>({})[]{(){{[{[([])]}]}}})`

Dataset size. We evaluate three training budgets: 400 steps (small), 600 steps (medium), and 1000 steps (large), using a batch size of 100. Results in the main text are reported under the medium budget, with additional results for other regimes presented in Appendix B.

Notes. (i) The task is strictly suffix completion (no in-filling in the middle). (ii) Unless stated otherwise, training and evaluation use exact-match on the m-token suffix.

A.3 Survo Dataset

Task. Given a partially filled $n \times n$ grid, fill the blanks in the top-left $(n{-}1) \times (n{-}1)$ block with integers so that each row sum (last column) and each column sum (last row) equals the sum of the other entries in that row/column. We report *exact-match* accuracy on all blank cells.

Construction. We follow SynLogic[3] and fix $n{=}4$. Entries in the top-left 3×3 block take values in $\{1, \ldots, 9\}$ (denoted $[\texttt{min\_num}, \texttt{max\_num}] = [1, 9]$). Let $G \in \mathbb{Z}_{\geq 0}^{n \times n}$ be the full grid and $\mathcal{M} \subseteq \{1, \ldots, n{-}1\}^2$ the set of blank locations. Constraints are:

$$\sum_{j=1}^{n-1} G_{ij} = G_{in} \quad (i = 1, \ldots, n{-}1), \qquad \sum_{i=1}^{n-1} G_{ij} = G_{nj} \quad (j = 1, \ldots, n{-}1),$$

and $G_{nn} = \sum_{i=1}^{n-1} G_{in} = \sum_{j=1}^{n-1} G_{nj}$. Training labels provide the fully completed grid G.

[2]https://github.com/MiniMax-AI/SynLogic/tree/main/games/tasks/dyck_language
[3]https://github.com/MiniMax-AI/SynLogic/tree/main/games/tasks/survo

Difficulty levels. We vary the number of blanks $x := |\mathcal{M}|$ while keeping n=4 and $[1, 9]$:

$$\begin{aligned} &\text{level 1: } (n{=}4,\ x{=}1),\\ &\text{level 2: } (n{=}4,\ x{=}2),\\ &\text{level 3: } (n{=}4,\ x{=}3),\\ &\text{level 4: } (n{=}4,\ x{=}4),\\ &\text{level 5: } (n{=}4,\ x{=}5). \end{aligned}$$

Examples. Blue numbers are givens; orange entries are blanks/filled values.

- **Level 1** (x=1)

  Input: $\begin{bmatrix} 2 & 5 & 5 & 12 \\ 6 & 1 & 2 & 9 \\ 4 & \mathbf{X} & 5 & 16 \\ 12 & 13 & 12 & 37 \end{bmatrix}$ Full target: $\begin{bmatrix} 2 & 5 & 5 & 12 \\ 6 & 1 & 2 & 9 \\ 4 & 7 & 5 & 16 \\ 12 & 13 & 12 & 37 \end{bmatrix}$
- **Level 2** (x=2)

  Input: $\begin{bmatrix} 3 & 1 & 8 & 12 \\ \mathbf{X} & 3 & 1 & 12 \\ 1 & \mathbf{X} & 4 & 13 \\ 12 & 12 & 13 & 37 \end{bmatrix}$ Full target: $\begin{bmatrix} 3 & 1 & 8 & 12 \\ 8 & 3 & 1 & 12 \\ 1 & 8 & 4 & 13 \\ 12 & 12 & 13 & 37 \end{bmatrix}$
- **Level 3** (x=3)

  Input: $\begin{bmatrix} \mathbf{X} & 1 & 2 & 9 \\ 6 & \mathbf{X} & 1 & 10 \\ 4 & 2 & \mathbf{X} & 8 \\ 16 & 6 & 5 & 27 \end{bmatrix}$ Full target: $\begin{bmatrix} 6 & 1 & 2 & 9 \\ 6 & 3 & 1 & 10 \\ 4 & 2 & 2 & 8 \\ 16 & 6 & 5 & 27 \end{bmatrix}$
- **Level 4** (x=4)

  Input: $\begin{bmatrix} 8 & \mathbf{X} & \mathbf{X} & 22 \\ 6 & 9 & 1 & 16 \\ 6 & \mathbf{X} & \mathbf{X} & 19 \\ 20 & 23 & 14 & 57 \end{bmatrix}$ Full target: $\begin{bmatrix} 8 & 6 & 8 & 22 \\ 6 & 9 & 1 & 16 \\ 6 & 8 & 5 & 19 \\ 20 & 23 & 14 & 57 \end{bmatrix}$
- **Level 5** (x=5)

  Input: $\begin{bmatrix} \mathbf{X} & 9 & \mathbf{X} & 14 \\ 3 & 7 & \mathbf{X} & 16 \\ \mathbf{X} & 2 & \mathbf{X} & 14 \\ 9 & 18 & 17 & 44 \end{bmatrix}$ Full target: $\begin{bmatrix} 3 & 9 & 2 & 14 \\ 3 & 7 & 6 & 16 \\ 3 & 2 & 9 & 14 \\ 9 & 18 & 17 & 44 \end{bmatrix}$

Dataset size. We evaluate three training budgets: 5000 steps (small), 10000 (medium), and 15000 steps (large), using a batch size of 100. Results in the main text are reported under the medium budget, with additional results for other regimes presented in Appendix B.

A.4 Training details

All models were trained on a single NVIDIA A100 GPU.

k-Parity We train a 2-layer MLP with one hidden layer of 256 `ReLU` units and a single-logit head. The objective is binary cross-entropy, optimized with Adam (learning rate 1×10^{-3}, weight decay 1×10^{-2}). Each update uses a batch of 1,000 examples drawn by a curriculum sampler over the five difficulty levels (uniform unless stated otherwise).

Dyck Language We fine-tune a 3-layer Transformer (hidden size 256) with supervised fine-tuning (SFT). Optimization uses AdamW (learning rate 5×10^{-5}, weight decay 0.1), gradient clipping via `max_grad_norm`$= 1.0$, and a *constant* learning-rate schedule (no decay, no warmup). Batching is controlled by a curriculum batch sampler with batch size 100.

Survo We fine-tune a 6-layer Transformer (hidden size 256) with the same optimizer, clipping, learning-rate schedule, and curriculum batching as in Dyck.

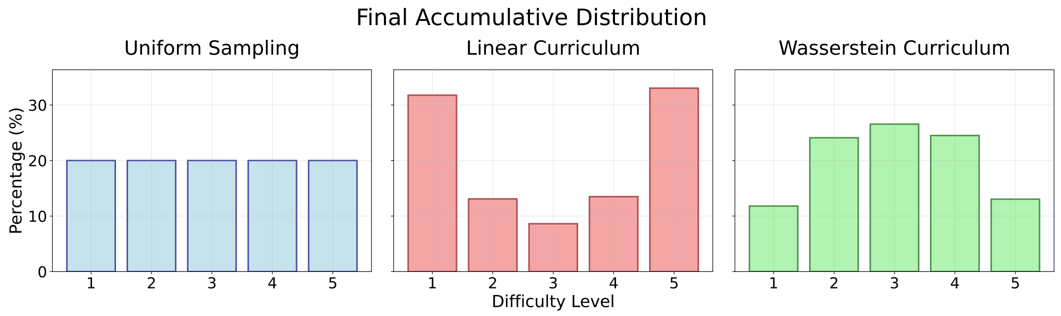


Figure 7: Final accumulative distributions under different curricula. **Left:** uniform sampling; **middle:** linear interpolation; **right:** Wasserstein interpolation (middle-focused).

A.5 Final Accumulative Distribution by Curriculum

Figure 7 shows the final accumulative distributions obtained at the end of training with speed parameter $\gamma = 1$ and temperature parameter $\tau = 1$. Uniform sampling yields a flat distribution, linear interpolation places more mass on the boundary levels (1 and 5), and Wasserstein interpolation concentrates mass on the middle levels (2–4). This illustrates that even when different curricula converge to similar overall performance, the Wasserstein curriculum achieves better sample efficiency, particularly at the higher levels.

B Additional Experimental Results

We provide additional results complementing Section 3.

B.1 Levelwise Learning Curves

We report levelwise learning curves corresponding to the experiments in Sections 3.1 and 3.2, across different dataset sizes. Here, the reverse curriculum swaps P_0 and P_1 from the Wasserstein and constant schedules. Figures 8–16 show the curves for small, medium, and large data regimes in the k-parity, Dyck, and Survo tasks.

Summary of findings.

1. **The effect of curriculum learning is task-dependent**, varying with the structure of each task and the relationships between difficulty levels.
2. **Reverse curricula consistently harm performance**, confirming the importance of progression direction.
3. **Curriculum effects are most pronounced in low-data regimes.** In large-data regimes, performance tends to converge across strategies, even without full saturation.
4. **Wasserstein curricula improve efficiency in small and medium regimes**, especially at higher levels, though gains can sometimes be modest.
5. **Linear interpolation skips intermediate levels without major harm in Dyck and Survo**, where even level 3 (with the least allocated mass) still improves along with other levels.
6. **Cumulative exposure is not sufficient:** the constant-matched baseline underperforms, showing that improvements come from the *path dynamics*, not just the final distribution.
7. **Stability varies across tasks:** in k-Parity, curricula induce sharper dips (due to XOR label flipping) whereas Dyck and Survo show smoother curves, suggesting that task sensitivity amplifies curriculum effects.

These findings are consistent with prior insights [14, 15], situating Wasserstein curricula within the broader class of curriculum learning methods that share these principles.

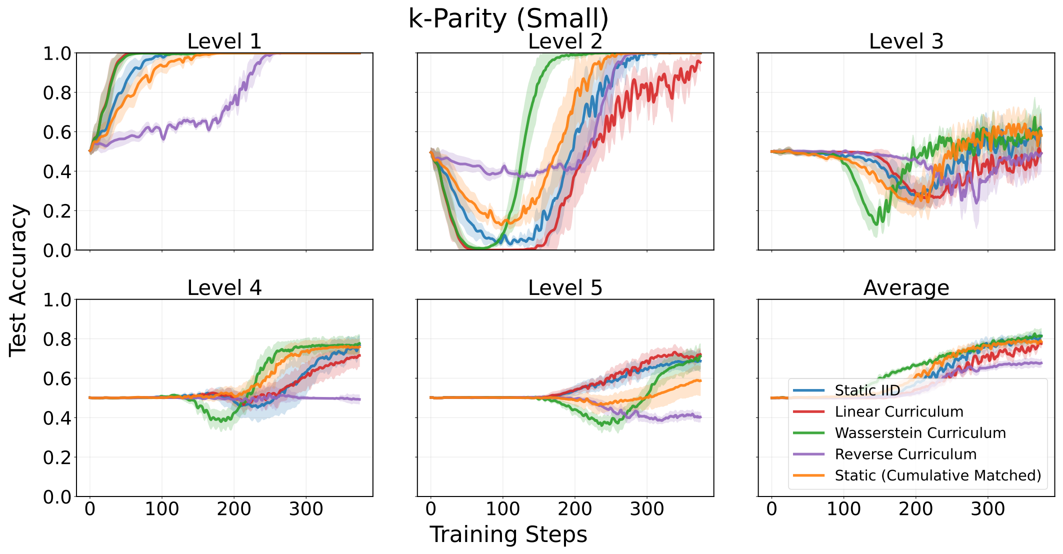


Figure 8: Levelwise learning curves for *k*-Parity (small data regime). Wasserstein curricula outperform uniform and linear, while reverse severely degrades performance.

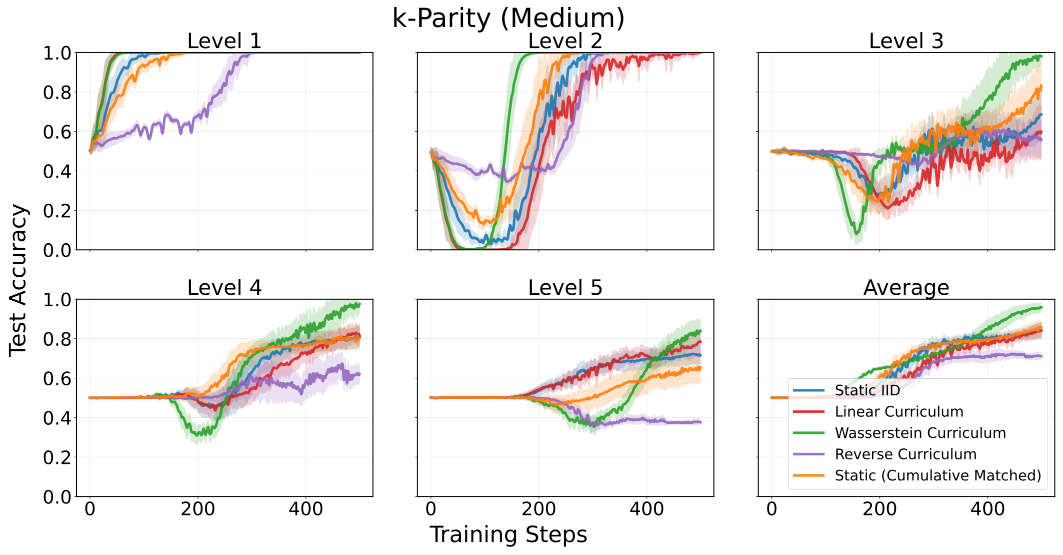


Figure 9: Levelwise learning curves for *k*-Parity (medium data regime). Wasserstein curricula continue to provide gains, while linear interpolation underperforms at the middle level (Level 3) due to its skipping behavior.

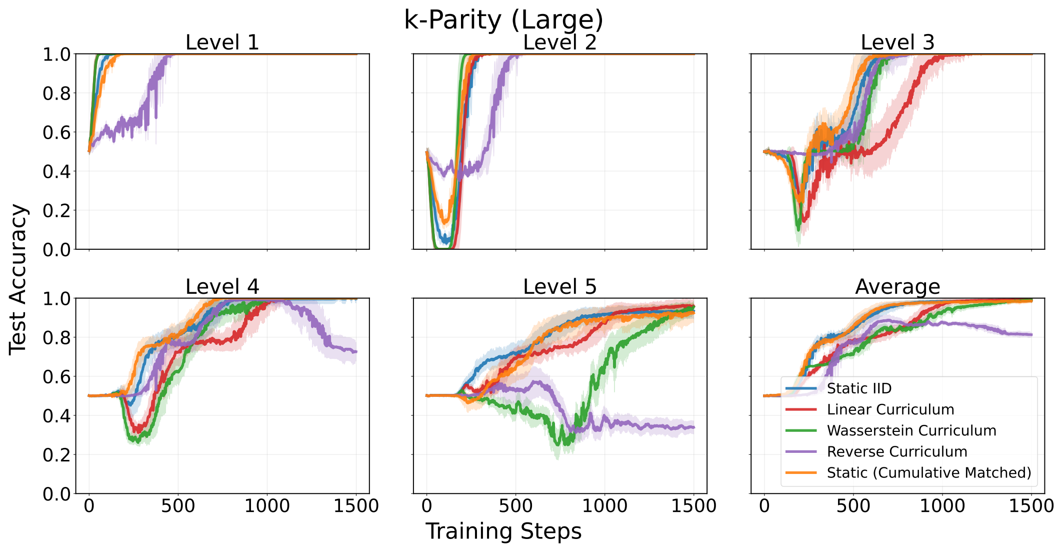


Figure 10: Levelwise learning curves for *k*-Parity (large data regime). Performance converges across curricula, indicating diminishing benefits of curriculum when data is plentiful.

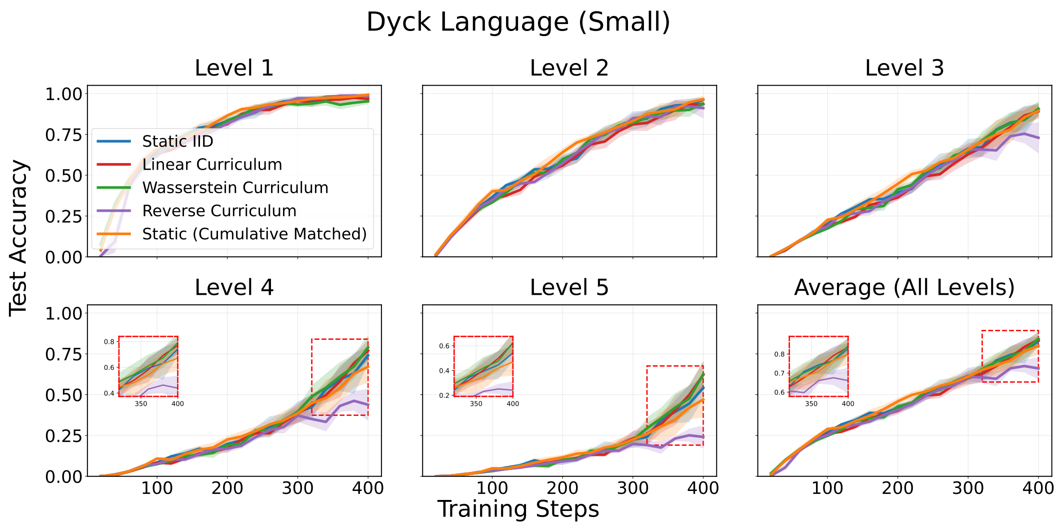


Figure 11: Levelwise learning curves for Dyck language (small data regime). Wasserstein improves stability and sample efficiency, while reverse is consistently harmful.

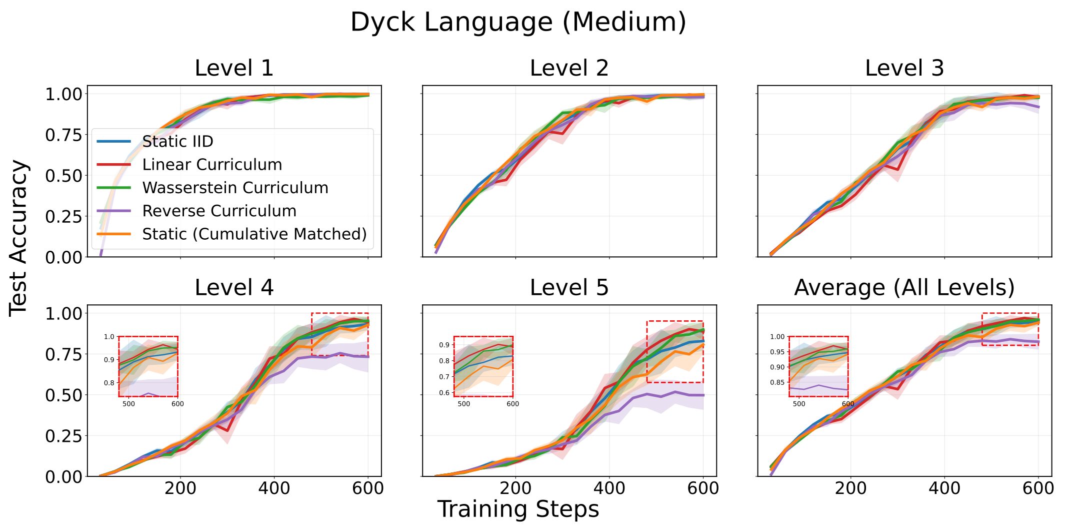


Figure 12: Levelwise learning curves for Dyck language (medium data regime). Differences between curricula shrink, though Wasserstein maintains slight advantages.

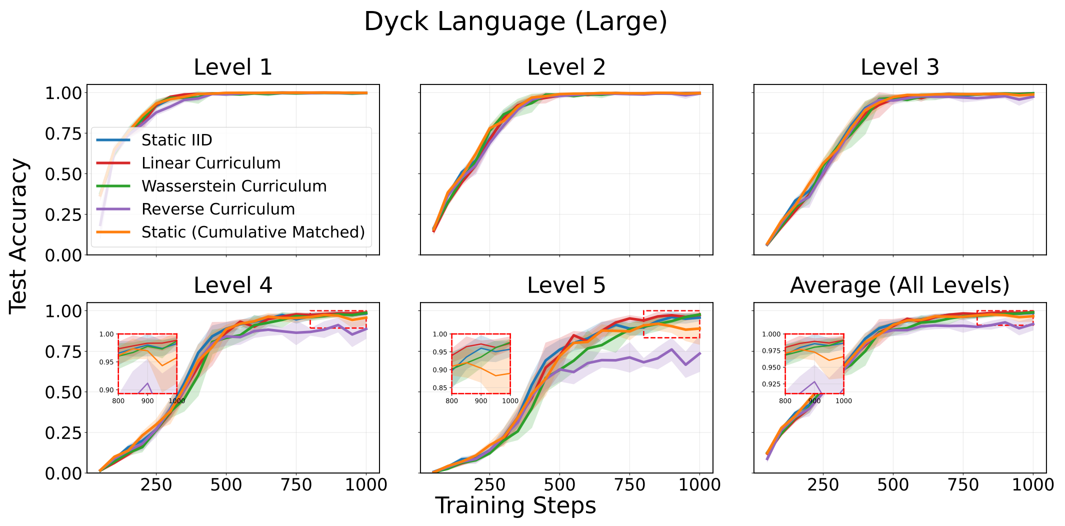


Figure 13: Levelwise learning curves for Dyck language (large data regime). All strategies converge to similar performance, showing little curriculum effect.

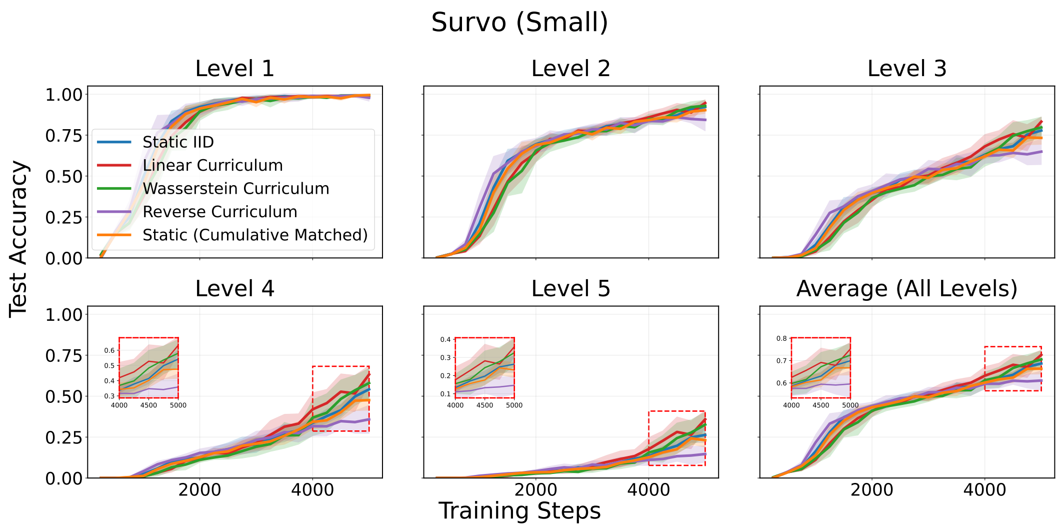


Figure 14: Levelwise learning curves for Survo (small data regime). Wasserstein improves learning at higher levels, while reverse again underperforms.

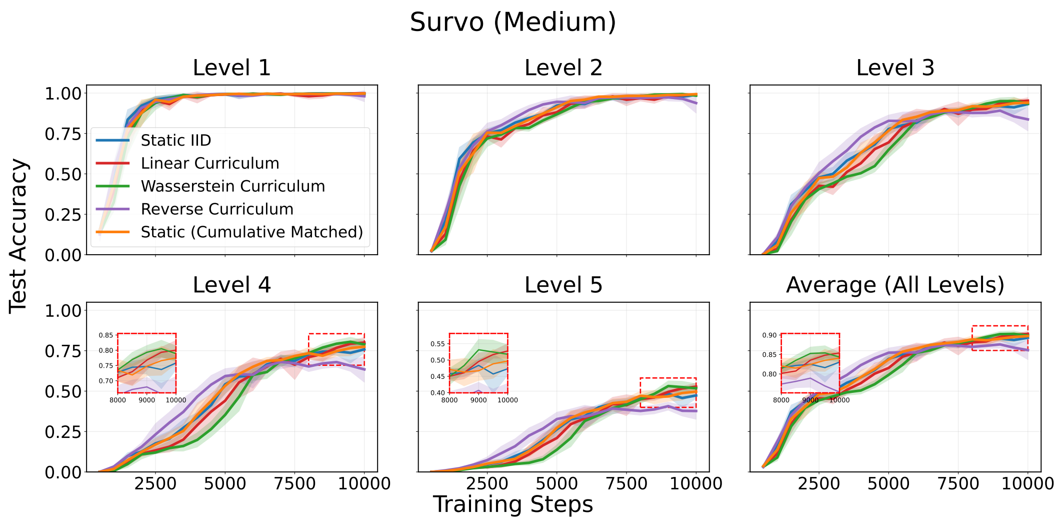


Figure 15: Levelwise learning curves for Survo (medium data regime). Wasserstein offers moderate gains, though overall trends across curricula begin to align.

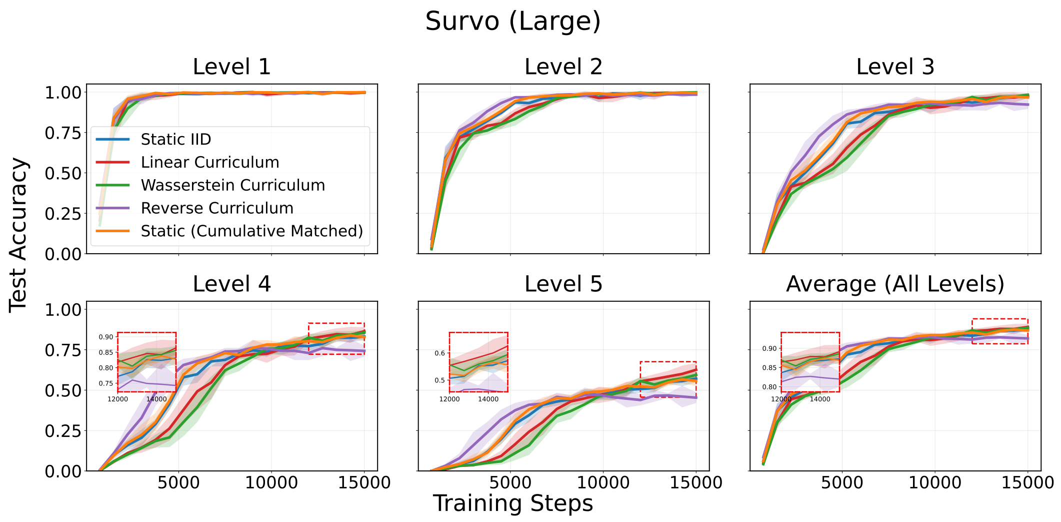


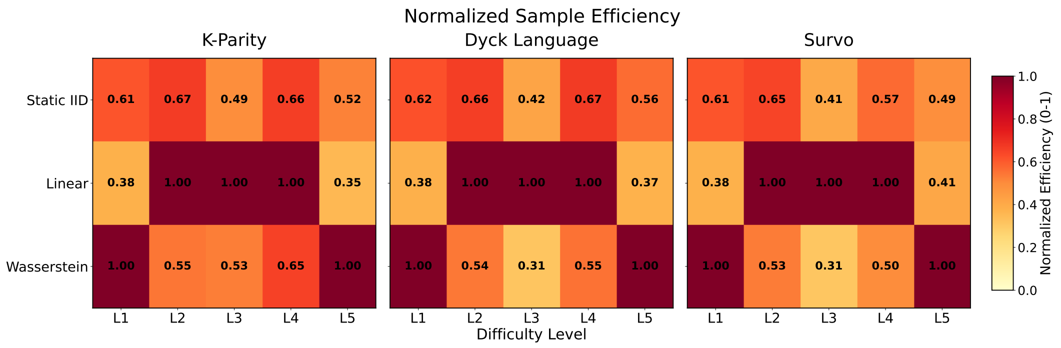
Figure 16: Levelwise learning curves for Survo (large data regime). Performance converges across curricula, indicating that curriculum effects diminish when data is abundant.

B.2 Sample Efficiency across Difficulty Levels

We study how curriculum learning affects sample efficiency at each difficulty level.

Setup. For level l, we define *sample efficiency* as the ratio between the final accuracy achieved at level l and the cumulative number of examples drawn from that level. We then normalize this quantity by dividing by the maximum efficiency attained among static i.i.d., linear interpolation, and Wasserstein interpolation curricula.

Findings. Figure 17 summarizes the results. Linear interpolation yields higher sample efficiency at intermediate levels, while Wasserstein interpolation excels at both the easiest and hardest levels. In practice, since data from harder regimes is often scarce, this suggests that Wasserstein-based curricula offer a cost-effective path for learning challenging tasks by allocating training more efficiently at the extremes of the difficulty spectrum.



K-Parity	L1	L2	L3	L4	L5
Static IID	0.61	0.67	0.49	0.66	0.52
Linear	0.38	1.00	1.00	1.00	0.35
Wasserstein	1.00	0.55	0.53	0.65	1.00

Dyck Language	L1	L2	L3	L4	L5
Static IID	0.62	0.66	0.42	0.67	0.56
Linear	0.38	1.00	1.00	1.00	0.37
Wasserstein	1.00	0.54	0.31	0.55	1.00

Survo	L1	L2	L3	L4	L5
Static IID	0.61	0.65	0.41	0.57	0.49
Linear	0.38	1.00	1.00	1.00	0.41
Wasserstein	1.00	0.53	0.31	0.50	1.00

Figure 17: Normalized sample efficiency across difficulty levels. Values are normalized within each task (k-Parity, Dyck, Survo) and levels.

B.3 Extended Results of Section 3.3

We provide additional speed parameter γ, temperature τ ablation experiments on the Dyck language and Survo tasks.

Setup. We sweep over temperatures $\tau \in \{0.1, 0.5, 1.0, 2.0, 10.0\}$ and speed parameters $\gamma \in \{0.1, 0.5, 1.0, 2.0, 10.0\}$, reporting average accuracy across levels.

Findings. Figure 18 summarizes the results. Overall, performance varies smoothly across the (τ, γ) grid outside the collapse region. For Dyck, low τ and low γ (top left corner) lead to sharp drops in accuracy, mirroring the k-Parity case: moving too quickly through the curriculum is harmful. The degradation stems from catastrophic forgetting, where early-level performance (Levels 1–2) collapses by the end of training. In contrast, Survo does not exhibit such forgetting. Here, faster progression (low γ or τ) can even improve performance, suggesting that the task structure tolerates or benefits from rapid exposure to harder levels. In the following sections, we analyze these failure cases and explain why Survo, unlike Dyck, remains resilient to forgetting.

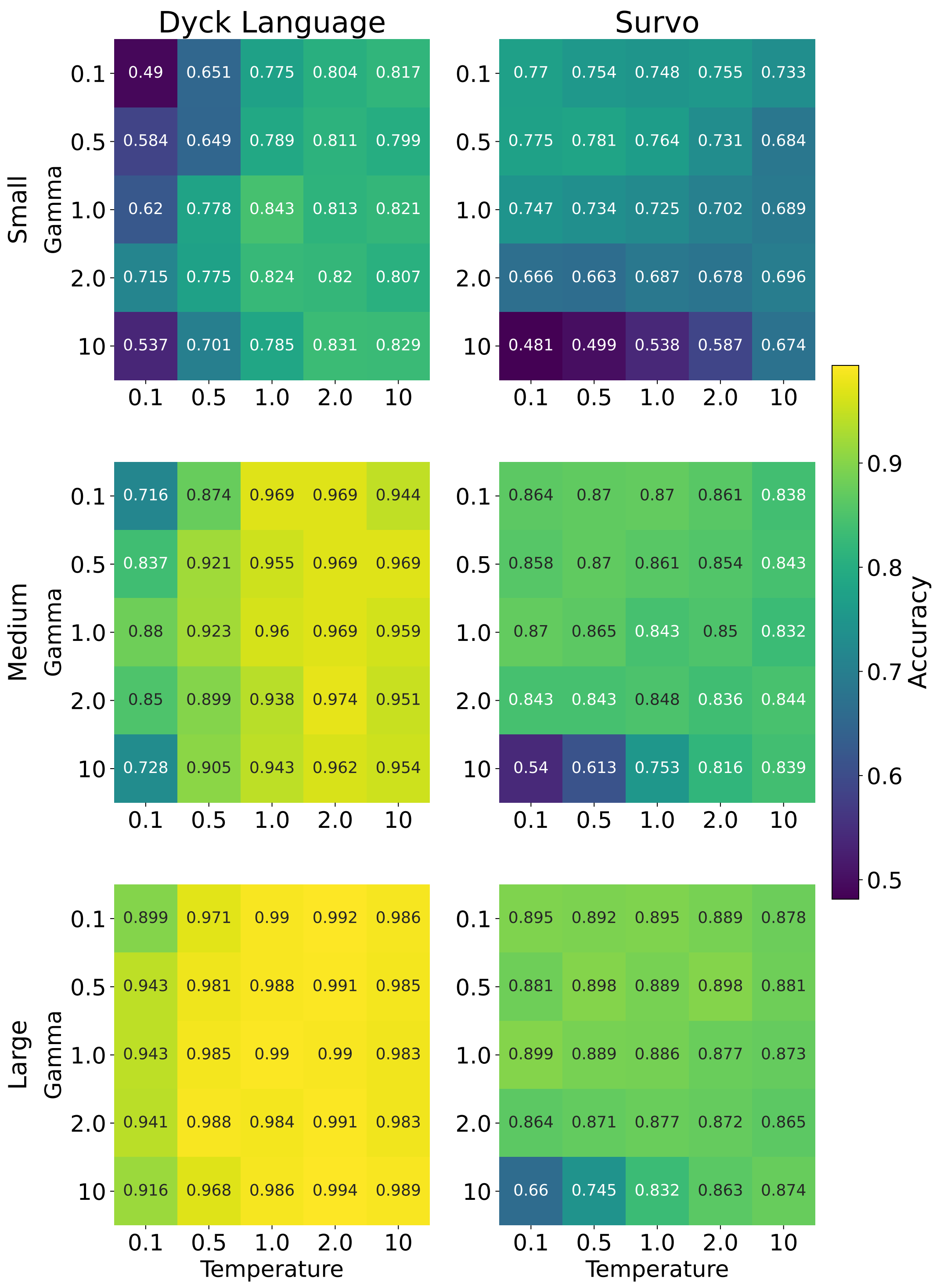


Figure 18: Temperature γ ablation results on Dyck language and Survo tasks. Dyck suffers from catastrophic forgetting in the low-τ, low-γ regime, while Survo remains stable.

B.4 Visualization of Failure Cases

We visualize three collapse cases across different values of the speed parameter γ and temperature τ, as identified in Section 3.3. All results are shown in the medium-data regime.

k-Parity ($\tau = 0.1, \gamma = 0.1$)**.** Figure 19 illustrates a representative failure case from the γ–τ analysis. Here, rapid progression to harder levels prevents the model from mastering any level. In particular, accuracy at early levels collapses and does not recover, indicating that progress on lower levels is necessary for success on harder ones. This highlights the dependency structure in k-Parity: learning at shallow depths provides critical scaffolding for deeper levels.

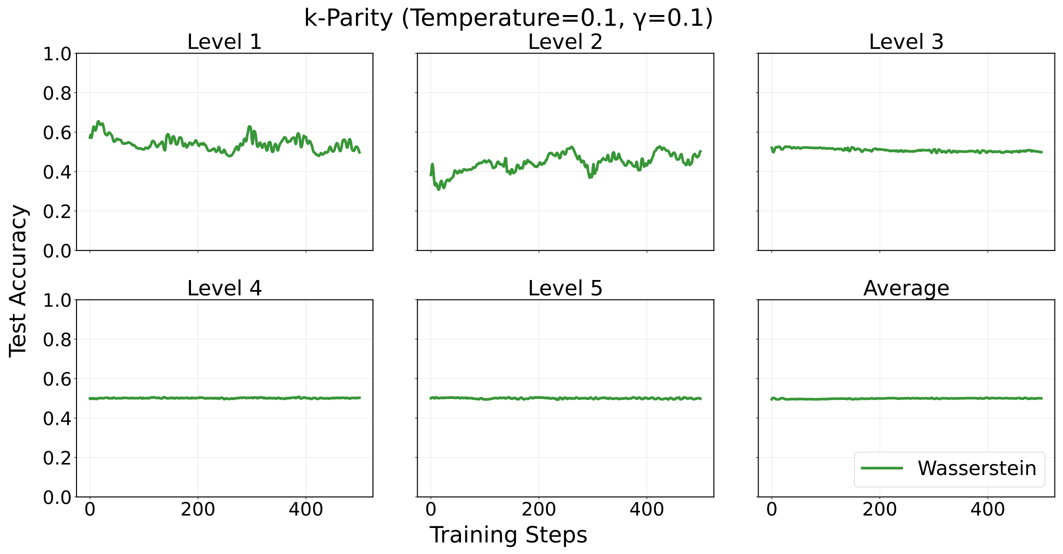


Figure 19: **k-Parity failure case.** Test accuracy across levels under $\tau = 0.1$ and $\gamma = 0.1$ in the medium-data regime. Rapid progression to harder levels prevents effective learning, leading to collapse across all levels.

Dyck Language ($\tau = 0.1, \gamma = 0.1$)**.** Figure 20 shows a failure case in the Dyck task. Accuracy on Levels 1–2 improves early in training but later collapses, a clear instance of catastrophic forgetting. This arises when early levels are removed too quickly, causing the model to overwrite previously learned skills.

Survo ($\tau = 0.1, \gamma = 10.0$)**.** Figure 21 shows a failure case in the Survo task. Although levels 1 to 2 are learned quickly and reach high accuracy, progress in Levels 4 to 5 remains minimal. This occurs because the curriculum lingers too long on easy levels, leaving insufficient exposure to harder ones. Proper pacing is therefore crucial for success.

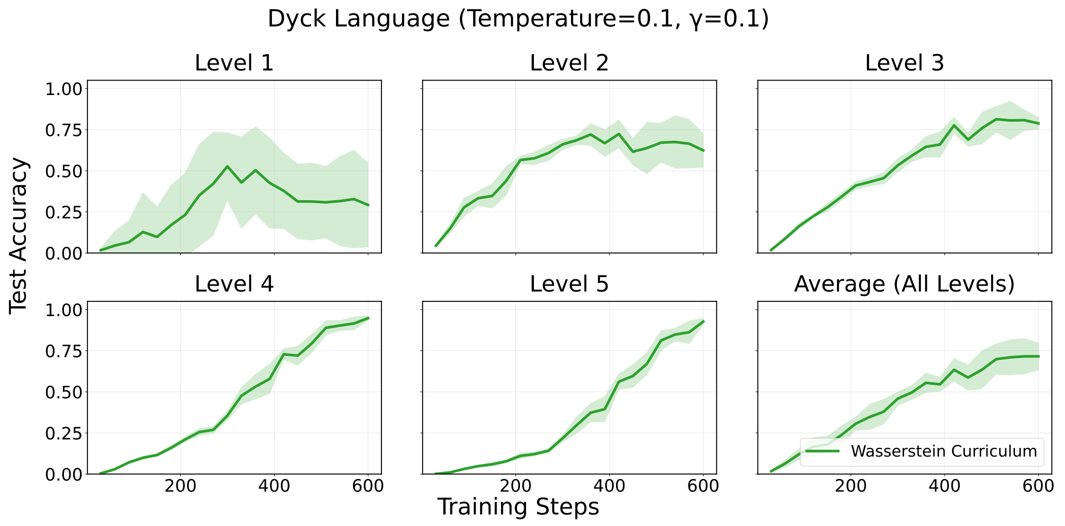


Figure 20: **Dyck Language failure case.** Test accuracy across levels under $\tau = 0.1$ and $\gamma = 0.1$ in the medium-data regime. Levels 1–2 collapse late in training, illustrating catastrophic forgetting.

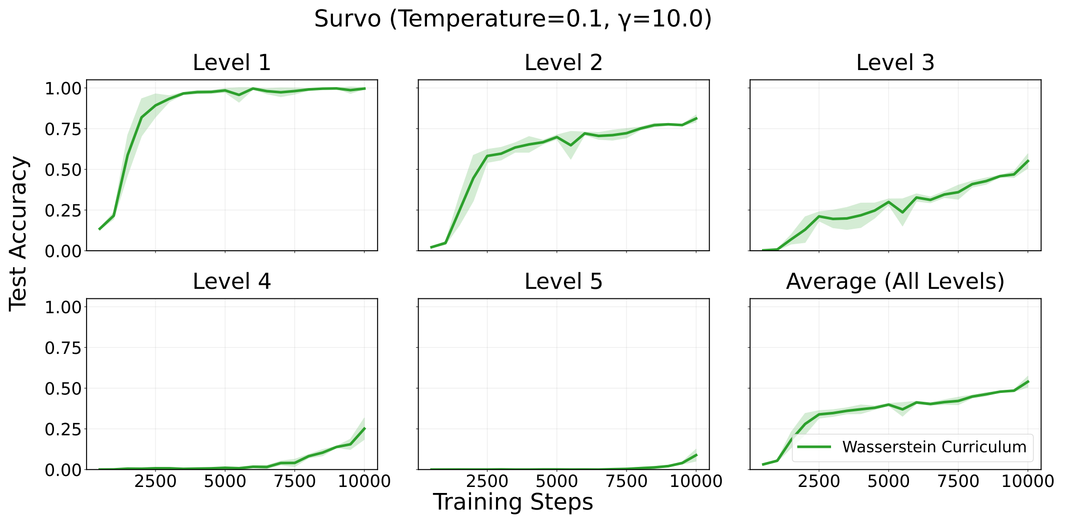


Figure 21: **Survo failure case.** Test accuracy across levels under $\tau = 0.1$ and $\gamma = 10.0$ in the medium-data regime. The model spends too long on easy levels and fails to acquire harder ones.

B.5 Why is Survo less sensitive to catastrophic forgetting and curriculum?

We hypothesize that Survo is less sensitive because learning at one level interacts only weakly with other levels.

Setup. To examine the hypothesis, we sample examples from each level, compute gradients of the LM head weights, and measure pairwise cosine similarities across levels. We then compare how these similarities evolve during training for Dyck Language and Survo.

Findings. Figure 22 shows the results. In Dyck, gradients remain strongly correlated across levels, so progress at one level easily interferes with others. This explains why Dyck suffers from catastrophic forgetting under low γ, low τ, and limited data for some levels: gradients from other levels overwrite previous progress. In contrast, Survo gradients quickly decorrelate, becoming nearly orthogonal as training advances. This weak coupling reduces cross-level interference, which prevents severe forgetting but also limits the potential benefits of curriculum—exposure to easier levels provides little leverage for learning harder ones. The effect is consistent across data regimes, curricula, and even when measuring gradients on layer normalization parameters. While this decorrelation explains why Survo is less affected by forgetting, the broader challenge of enabling hard-to-easy generalization in Survo remains open.

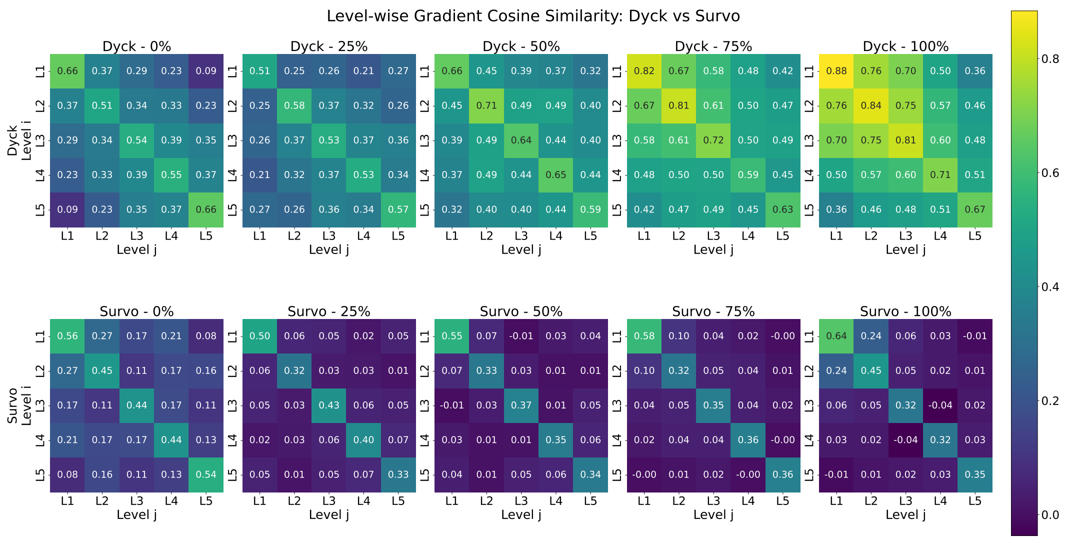


Figure 22: Evolution of gradient cosine similarity across levels during training (medium data). Each column shows training progress at 0%, 25%, 50%, 75%, and 100%. Top row: Dyck Language, where gradients remain correlated across levels, making the task sensitive to curriculum and prone to forgetting. Bottom row: Survo, where gradients decorrelate over time, reducing interference but also limiting the usefulness of curriculum.