Curriculum Learning as Transport: Understanding Curricula with Wasserstein Geodesics

Changho Shin*
Princeton University
Princeton, NJ, USA
cs1095@princeton.edu

David Alvarez-Melis
Microsoft Research & Harvard University
Cambridge, MA, USA
daalvare@microsoft.com

Abstract

Curriculum learning is governed by several coupled design choices—how difficulty is defined, how examples are ordered, how much exposure each level receives, and how quickly training moves across levels—making it hard to isolate what actually helps. We present *Wasserstein curriculum paths*, a simple transport-based framework that decouples these factors by representing curricula as trajectories of training distributions over discrete difficulty levels. Across a calibrated synthetic suite with 12 tasks and 33 difficulty axes, we use this framework to isolate the effects of ordering, matched exposure, endpoint smoothness, and pacing under fixed training budgets. We find that curriculum effects are strongly context-dependent: no single strategy dominates across tasks, difficulty axes, and budgets, and curricula mainly change where a fixed budget is spent most effectively. Within this framework, easy-to-hard ordering improves hard-level performance relative to exposure-matched static sampling, showing that the benefit is not explained by cumulative exposure alone. We further show that endpoint smoothness and pacing substantially affect where along the difficulty spectrum a curriculum is effective. Finally, we show that the same transport view naturally supports extensions to learned pacing through geometry and to structured difficulty spaces beyond one-dimensional orderings.

1 Introduction

Curriculum learning aims to improve training by presenting examples in a structured order, often from easier to harder instances (Bengio et al., 2009). But in practice, curriculum effects depend on several coupled design choices: how difficulty is defined, whether training progresses easy-to-hard or hard-to-easy, how much total exposure each level receives, and how quickly the curriculum moves across levels (Hacohen & Weinshall, 2019; Wu et al., 2021; Jia et al., 2026). As a result, even when a curriculum helps, it is often unclear why: are the gains due to ordering itself, greater exposure to easier examples, smoother transitions between levels, or a better pacing of a fixed training budget?

Prior curriculum methods—including self-paced selection, teacher-student schedulers, bandit approaches, and competence-based pacing—often improve performance (Kumar et al., 2010; Matiisen et al., 2020; Graves et al., 2017; Platanios et al., 2019), but they usually change several design choices at once, making it hard to isolate their contributions. Recent studies also ask more directly when ordering helps (Wu et al., 2021; Jia et al., 2026), but they do not ultimately provide a common framework for separating ordering from matched exposure, endpoint choice, pacing, and geometry within one controlled setup.

We propose Wasserstein interpolation as a unified controlled framework for studying curriculum design. By viewing a curriculum as a path of training distributions over ordered difficulty levels, Wasserstein paths provide a smooth way to move mass through nearby levels between easy-heavy and hard-heavy training distributions (McCann, 1997; PeyrÃ© &

*Work done during a research internship at Microsoft Research.

Cuturi, 2019). This lets us vary one design choice at a time while holding the others fixed, enabling controlled comparisons of ordering, matched exposure, endpoint smoothness, pacing, and geometry within a single parameterization.

Using a calibrated synthetic suite spanning 12 tasks and 33 difficulty axes, we use this framework to isolate which components of curriculum design matter and under what conditions. Three main findings emerge:

Curriculum effects are context dependent. Across the synthetic suite, no single curriculum dominates across tasks, difficulty axes, and budgets. Instead, curricula mainly shift where a fixed training budget is used most efficiently across difficulty levels, with smooth easy-to-hard progression particularly efficient on the hardest levels.

Ordering matters beyond cumulative exposure. Easy-to-hard progression improves hard-level performance relative to an exposure-matched static baseline, whereas hard-to-easy ordering often hurts, indicating that timing matters, not just cumulative exposure.

Pacing and endpoint design matter. Even within an easy-to-hard curriculum, the same path can help or hurt depending on how sharply the endpoint distributions are concentrated and how quickly the path is traversed. Very sharp endpoints are brittle under either overly aggressive or overly conservative pacing.

Beyond these main findings, we showcase how the framework can adapt pacing through the underlying geometry and extend curriculum paths from one-dimensional orderings to structured difficulty spaces.

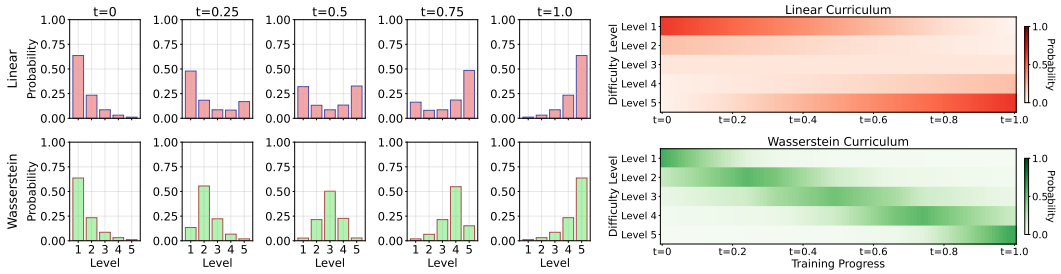
Taken together, these results establish Wasserstein curricula as a principled framework for studying curriculum components, understanding when they matter, and extending curriculum design to adaptive pacing and structured difficulty spaces.

2 Related Work

We review the most relevant prior work here and provide further discussion in Appendix A.

Curriculum learning and difficulty-aware training. Curriculum learning studies whether organizing training examples by difficulty can improve optimization and generalization (Bengio et al., 2009). Automated variants choose what to show and when through self-paced, teacher–student, bandit, and competence-based schedules (Kumar et al., 2010; Matiisen et al., 2020; Graves et al., 2017; Platanios et al., 2019), while training dynamics and related signals have been used to estimate example difficulty (Hacohen & Weinshall, 2019; Swayamdipta et al., 2020; Shrivastava et al., 2016; Toneva et al., 2019). Wang et al. (2022) organize this literature around difficulty measurement and training scheduling, and Meng et al. (2026) combine these components in a unified dynamic framework. Recent work also asks more directly when ordering helps: Wu et al. (2021) find clearer gains under limited budgets or label noise, while Jia et al. (2026) show in LLM math post-training that the preferred direction depends on model capability, task complexity, and the difficulty metric. Our work studies a different question: given a difficulty structure, which aspects of the resulting curriculum matter? We represent curricula as paths over distributions on discrete difficulty levels and separate the difficulty geometry, endpoints, direction, cumulative exposure, and pacing. This enables controlled comparisons of factors that are often varied together, while also supporting extensions to adaptive geometry and structured difficulty spaces.

Optimal Transport Optimal transport (OT) provides a geometry for comparing probability distributions based on the cost of transporting mass between them (Villani, 2008). In this geometry, Wasserstein interpolation defines geodesic paths that smoothly transform one distribution into another (McCann, 1997; PeyrÃ© & Cuturi, 2019; Santambrogio, 2015). Prior OT-based curriculum work, especially in reinforcement learning, uses these ideas to construct or sequence tasks by generating intermediate training stages between task or environment distributions (Huang et al., 2022; Klink et al., 2022). In contrast, we use Wasserstein interpolation as an analytical framework over distributions on a fixed difficulty



(a) Linear vs. Wasserstein interpolation. (b) Curriculum heatmap.

Figure 1: Comparison of linear and Wasserstein curricula. (a) Wasserstein interpolation moves probability mass progressively through the level geometry, whereas linear interpolation directly mixes the endpoint distributions. (b) Viewing P_t as the batch-level sampling distribution over time shows that Wasserstein places substantially more mass on intermediate levels during the transition.

axis, allowing us to cleanly separate ordering, exposure, pacing, and endpoint effects within a single parameterization. This perspective focuses less on designing new curricula and more on isolating which components of curriculum design drive observed gains.

3 Preliminaries and Experimental Setup

We first introduce Wasserstein-geodesic curricula, then summarize the common protocol that turns them into a controlled study of curriculum design.

3.1 Wasserstein–Geodesic Curriculum

For each task, we partition the training data into L ordered difficulty levels. At training step k, each batch is sampled from a probability vector $P^k \in \Delta^{L-1}$, where P^k_ℓ is the probability of drawing from level ℓ. A curriculum is therefore a sequence of sampling distributions over these levels. We represent it as a continuous path $\{P_t\}_{t\in[0,1]}$ between endpoint distributions $P_0, P_1 \in \Delta^{L-1}$, together with a schedule $t_k \in [0,1]$ that maps training step k to a position on the path. The path specifies which mixtures of difficulty levels are visited, and the schedule specifies how quickly training moves through them, with $P^k = P_{t_k}$.

Wasserstein interpolation. We construct the path on the default one-dimensional ordering of levels: for $t \in [0,1]$,

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

where W is the Wasserstein distance on that fixed level geometry. This moves probability mass smoothly through adjacent levels between the two endpoint distributions.

For comparison, the linear interpolation baseline directly mixes the same endpoints,

$$P_t^{\text{lin}} = (1-t)P_0 + tP_1.$$

As illustrated in Figure 1, linear interpolation mixes the endpoint distributions directly, whereas the Wasserstein path progresses through intermediate levels, producing a more natural curriculum.

Curriculum with Wasserstein interpolation. By setting $P^k = P_{t_k}$, we obtain curricula that move mass through nearby levels. The choice of endpoints (P_0, P_1) determines where the curriculum starts and ends, the geometry determines which levels are treated as nearby, and the schedule $\{t_k\}$ controls how quickly training moves along the path. This makes Wasserstein curricula useful as a study lens: once the task axis is fixed, they expose several curriculum factors that can be varied cleanly, including endpoint choice, pacing, and

geometry, while preserving a natural progression over the same levels. Later sections use this structure to study endpoint families and traversal rates, learned geometry in WARP, and structured difficulty spaces beyond one-dimensional orderings.

3.2 Common Experimental Setup

We next introduce the common setup for our experimental suite. Unless varied explicitly in later sections, we keep the task construction, models, sampling protocol, and budget calibration fixed so that the suite can be used to study ordering, matched exposure, pacing, and geometry under shared conditions. We use this calibrated synthetic suite for the main study because it supports controlled comparisons; in Appendix D, we show that in real-data SFT settings with pretrained models and noisier difficulty labels, curriculum effects are much weaker and harder to disentangle.

Tasks and difficulty control. Unless otherwise stated, we study a synthetic suite of 12 tasks, adapted in part from SynLogic (Liu et al., 2025a). For each task, we construct ordered difficulty levels by varying one difficulty axis at a time, that is, one task attribute used to order examples from easier to harder while keeping the task semantics fixed. Calibrating and validating these within-task axes is part of the contribution, since it lets us compare curricula on controlled progressions rather than heterogeneous mixtures of examples. In the main study, this yields 33 task-by-difficulty-axis conditions, because several tasks contribute multiple difficulty axes. Appendix B.1 gives the full task list, examples, and exact level constructions.

Models. We use decoder-only Transformers (Vaswani et al., 2017) throughout, with model size matched to task complexity. Most tasks use small models trained from scratch, while language-like tasks fine-tune SmolLM-135M (Ben Allal et al., 2024). Model details are reported in Appendix B.2.

Training protocol. At training step k, mini-batches are sampled from a distribution $P^k \in \Delta^{L-1}$ over difficulty levels. We use *static i.i.d.* for the non-curriculum baseline, where batches are drawn independently from the same fixed distribution throughout training; by default, that distribution is uniform over difficulty levels. Unless varied explicitly, our default easy-to-hard runs use an easy-heavy start distribution P_0 and a hard-heavy end distribution P_1, where $(P_1)_\ell \propto \exp(\ell/\tau)$ and P_0 is the symmetric reversal of P_1, i.e., $(P_0)_\ell = (P_1)_{L-\ell+1}$. We use default temperature $\tau = 1$ and default schedule $t_k = k/T$.

Training budgets. Training budget is a key variable in curriculum learning (Wu et al., 2021). Rather than fixing one global step count, we calibrate small, medium, and large regimes separately for each benchmark from pilot static i.i.d. learning curves, so that the static i.i.d. baseline spans early, intermediate, and later stages of learning on that task. This keeps budget comparisons aligned across tasks instead of comparing one task at an early stage of learning and another at a much later stage. Exact budget values are listed in Appendix B.1, with the associated model and training details in Appendix B.2.

4 When and How Does Curriculum Learning Help?

We first study whether curriculum learning helps reliably or only in specific regimes. For this, we compare *static i.i.d.*, linear, and Wasserstein curricula across the 33 synthetic-suite task-by-difficulty-axis conditions in the main study, and report both overall and hardest-level accuracy.

4.1 When Curricula Help Is Highly Context Dependent

Setup. For each task-by-difficulty-axis condition at a given budget, we first average accuracy over 10 random-seed repetitions, then aggregate those condition-level means within each budget block. Each block contains 33 conditions.

Curriculum	Mean overall (%)	Mean hardest (%)	Overall wins	Hardest wins
Small budget				
Static i.i.d.	**56.6**	39.9	**13**	6
Linear	55.4	**46.4**	7	12
Wasserstein	**56.6**	46.3	**13**	**15**
Medium budget				
Static i.i.d.	79.2	66.1	11	5
Linear	78.8	**72.6**	10	**16**
Wasserstein	**79.7**	72.1	**12**	12
Large budget				
Static i.i.d.	92.7	86.0	6	1
Linear	92.9	89.6	12	13
Wasserstein	**93.2**	**89.7**	**16**	**19**

Table 1: Absolute summary for the three basic curricula, stratified by training budget. Each block contains 33 task-by-difficulty-axis conditions. The win columns denote the number of conditions in which a curriculum performs best under the corresponding accuracy metric. "Hardest" denotes the last available level for each benchmark.

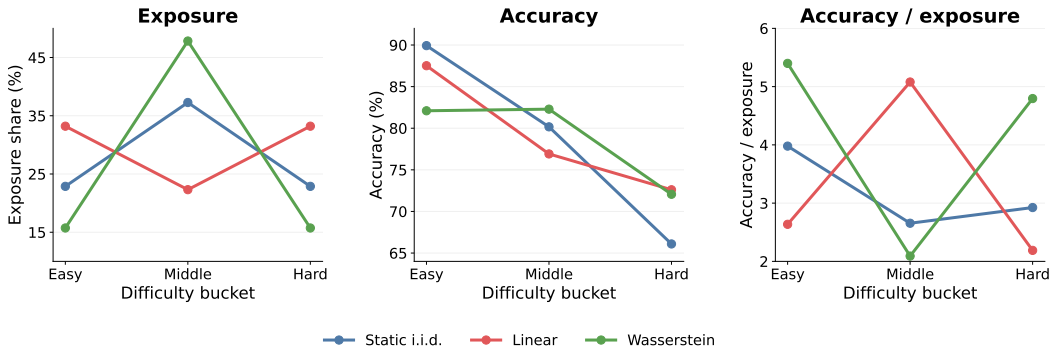


Figure 2: Exposure, accuracy, and exposure-adjusted accuracy under the medium budget. **Left:** cumulative exposure share over easiest, middle, and hardest probe buckets. **Middle:** final bucket accuracy on the same buckets. **Right:** exposure-adjusted accuracy on the same buckets. The middle bucket is the single middle level when the number of levels is odd and the merged middle pair when it is even, so these three probe buckets are not an exhaustive partition for five-level tasks. All panels compare only *static i.i.d.*, *linear*, and *Wasserstein*. The corresponding profiles across all three budgets are shown in Figure 21 in Appendix C.2.

Results. Table 1 reports the resulting summary separately for the small, medium, and large regimes. No single curriculum dominates the suite. All three methods win substantial subsets of conditions, and the ranking shifts with budget rather than settling on a universal best choice. The main lesson is therefore contextual rather than universal: whether curriculum helps depends jointly on task structure, the difficulty axis, and the available budget. Appendix C.1 gives the corresponding task-wise final-accuracy breakdowns.

Finding 1. *Curriculum effectiveness depends strongly on task, difficulty axis, and budget; no single strategy dominates.*

4.2 Curricula Shift the Data-Efficiency Profile Across Difficulty

Similar overall accuracy means can hide very different learning profiles. We therefore compare curricula not only by final accuracy, but also by where they are data-efficient across difficulty levels.

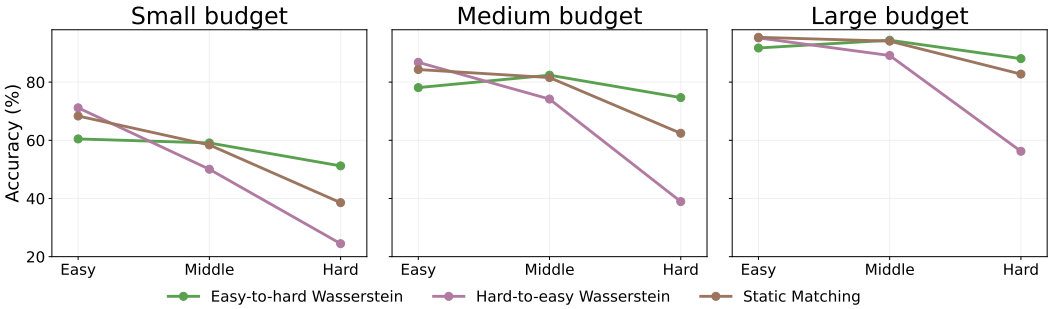


Figure 3: Ordering within the Wasserstein comparison. Each panel shows final accuracy on the easy, middle, and hard buckets for one training budget, corresponding to the first, middle, and hardest levels in the shared summary used here. "Static Matching" denotes the exposure-matched static baseline. Easy-to-hard Wasserstein consistently yields a flatter profile and stronger hard-bucket performance than either the static baseline or hard-to-easy Wasserstein.

Setup. Using the same setup as Section 4.1, Figure 2 summarizes medium-budget behavior on the easiest, middle, and hardest probe buckets. Exposure is the cumulative share of training mass assigned to each bucket over the full run. As a simple proxy for data efficiency, we use *exposure-adjusted accuracy*: final bucket accuracy divided by cumulative exposure to that bucket.

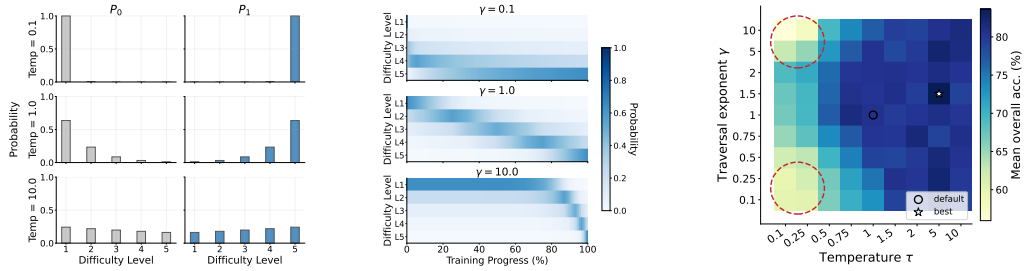
Results. Figure 2 shows that with a fixed budget, the main difference between curricula is not a uniform gain in final accuracy, but a different data-efficiency profile across difficulty. *Static i.i.d.* remains strongest in final accuracy on the easiest bucket. Linear is strongest in exposure-adjusted accuracy in the middle bucket, while Wasserstein, which follows a natural smooth progression over the ordered levels, has the highest exposure-adjusted accuracy on the hardest bucket. This pattern is not driven by a few outliers: on the hard bucket, Wasserstein has higher exposure-adjusted accuracy than linear in all 99 task-by-axis-and-budget conditions. Figure 21 in Appendix C.2 shows that the same qualitative pattern persists across the small, medium, and large budgets. The main effect of curriculum is therefore to reallocate where a fixed budget pays off, with Wasserstein especially efficient on the hardest bucket, which may matter when hard examples are scarce or costly. Appendix C.2 gives the corresponding budget-wise profile view, Appendix C.3 shows the task-wise final level profiles, and Appendix C.6 revisits the hard-bucket advantage through controlled bridge-effect experiments.

Finding 2. *With a fixed budget, curricula reshape the data-efficiency profile across difficulty; Wasserstein is especially efficient on the hardest bucket.*

5 Does Ordering Matter?

Section 4.2 showed that curricula with the same budget can induce different exposure and accuracy profiles across difficulty. We now ask whether ordering itself matters beyond cumulative exposure, comparing the default easy-to-hard Wasserstein path, the exposure-matched static baseline, and hard-to-easy Wasserstein on the same suite.

Setup. Using the same default setup as in the previous sections, we compare easy-to-hard Wasserstein, the default ordering used throughout the paper, with two controls that keep cumulative exposure fixed while changing ordering. The exposure-matched static baseline, shown as "Static Matching" in Figure 3, preserves cumulative exposure to each level but removes the temporal progression, so it uses the same fixed batch-sampling distribution throughout training. Hard-to-easy Wasserstein traverses the same path family in reverse,



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

Figure 4: **(a)** Lower τ sharpens the endpoints; larger τ smooths them. **(b)** γ changes how fast training moves along the same easy-to-hard path. **(c)** Medium-budget sweep over 20 task-by-difficulty-axis conditions and three seeds. A broad region with moderate pacing and smoother endpoints performs best; the red dashed circles highlight collapse under very sharp endpoints with either overly aggressive or overly conservative pacing.

and therefore preserves the same cumulative exposure while reversing the direction of progression.

Results. Figure 3 shows that neither control recovers the easy-to-hard profile. The exposure-matched static baseline retains more strength on the easy bucket, but it misses the hard-bucket gains of easy-to-hard Wasserstein even when their overall averages remain similar. Hard-to-easy Wasserstein is more damaging still and often harms performance outright: it shifts strength toward the easy end and away from the hard bucket, where curricula matter most. This effect is broad rather than isolated to a few tasks: on the hardest level, forward Wasserstein exceeds reverse Wasserstein in 92 of 99 task-by-axis-and-budget conditions. The benefit of easy-to-hard progression therefore does not come from cumulative exposure alone; it comes from when examples are seen during training. Appendix C.5 studies these directional transfer patterns more directly with level-to-level transfer experiments.

Finding 3. *Ordering matters beyond cumulative exposure: exposure-matched static sampling misses the hard-bucket gains of easy-to-hard progression, and hard-to-easy ordering often harms.*

6 Pacing and Smoothness Matter

Even within an easy-to-hard curriculum, do pacing and endpoint smoothness still matter? We vary endpoint smoothness (τ) and traversal speed (γ) along the same Wasserstein path.

Setup. We use seven tasks with short enough runtimes to make the full grid sweep feasible, giving 20 task-by-difficulty-axis conditions. Throughout this subsection we keep the same easy-to-hard Wasserstein path

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

and vary only its endpoint smoothness and pacing. The endpoint family and traversal are

$$(P_1)_\ell \propto \exp(\ell/\tau), \qquad (P_0)_\ell = (P_1)_{L-\ell+1}, \qquad t_k = (k/T)^\gamma.$$

Here ℓ indexes difficulty levels, L is the number of levels, k is the training step, and T is the total number of training steps. Smaller τ makes the endpoints sharper and larger τ smooths them, while γ controls how quickly training moves along the path: smaller values move training toward harder levels sooner, while larger values keep training longer on easier and intermediate levels. We sweep a 9×9 grid over $\tau, \gamma \in \{0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 5.0, 10.0\}$ in the medium-budget regime only, with three seeds per condition.

Figure 5: Geometry controls pacing even when endpoints stay fixed. Left: standard and an affinely rescaled learned geometry on the same level axis. Right: the resulting easy-to-hard Wasserstein trajectories. Stretching the early-to-mid region and compressing the hard end makes the path dwell longer before a sharper final handoff.

Results. Figure 4(c) reveals a broad good region rather than a single best setting. Moderate pacing and smoother endpoints work well, while very sharp endpoints are brittle under both overly aggressive and overly conservative pacing. The default setting already lies in a strong region, but it is not uniquely optimal. Easy-to-hard ordering alone is therefore not enough: the same path can help or hurt depending on smoothness and pacing.

Finding 4. *Pacing and endpoint smoothness matter: very sharp endpoints can fail under either overly aggressive or overly conservative pacing.*

7 Adaptive Geometry as Learnable Pacing

If pacing matters, the next question is how to adapt it without leaving the Wasserstein framework. We do so indirectly through geometry: because Wasserstein paths depend on the underlying distances between levels, changing that geometry changes pacing while preserving the same easy-to-hard endpoints. As a prototype, we instantiate this idea as WARP (Wasserstein-Adaptive Reshaping of Paths), which learns a geometry once after a short warmup and then follows the same forward Wasserstein path on the learned locations. Appendix B.4 gives the full definition; here we only highlight the learned edge-length rule.

Setup. After a short uniform warmup, WARP probes gradient alignment between neighboring levels and converts adjacent cosine couplings into relative edge lengths, so higher similarity shortens an edge and lower similarity lengthens it.

Results. Table 2 shows that even this first prototype beats fixed Wasserstein at small and medium budgets, while matching it at large budget. This proof-of-point result shows that adapting the geometry can improve pacing along the same easy-to-hard path.

Table 2: Mean overall accuracy (%) by budget.

Method	Small	Medium	Large
Wasserstein	56.8	79.8	92.9
WARP	**57.1**	**80.6**	92.9
Gain	+0.2	+0.9	+0.0

8 Structured Difficulty Spaces Beyond One-Dimensional Orderings

So far, we have treated difficulty as a one-dimensional ordering. Here we ask whether the same framework still works when levels are connected by a richer structure. To illustrate this, we construct two structured difficulty spaces from the arithmetic-operator family, a cube and a tree, and compare *static i.i.d.*, *Wasserstein*, and WARP.

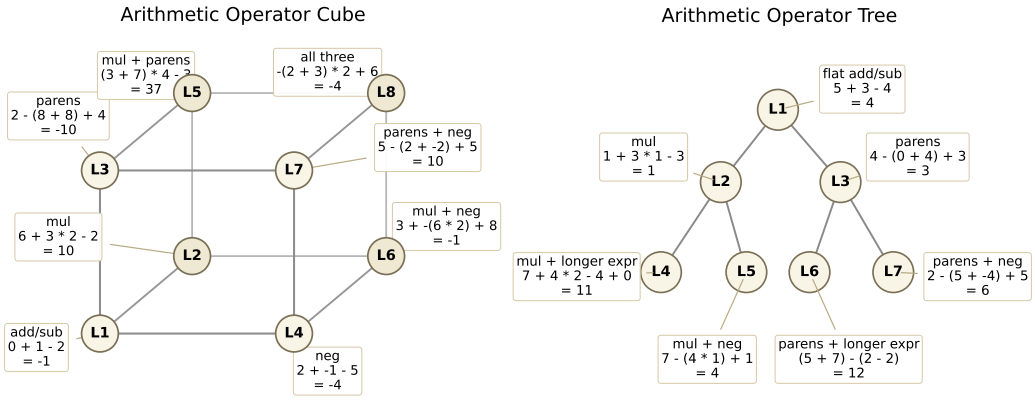


Figure 6: Structured benchmarks, a cube and a tree, built from arithmetic operators. Nodes are labeled by level, and nearby callouts show example expressions.

Cube

Budget	Nodes	Static i.i.d.	Wasserstein	WARP
Small	All	61.4	**62.2**	59.3
	Init	**82.9**	77.3	70.7
	Mid	60.8	**61.3**	59.2
	End	43.7	**53.0**	48.4
Medium	All	**77.6**	75.1	75.6
	Init	**95.0**	87.6	91.4
	Mid	**77.5**	74.5	75.7
	End	60.5	**66.3**	59.1
Large	All	**93.3**	93.2	90.5
	Init	**99.4**	98.4	96.6
	Mid	**93.6**	93.1	90.5
	End	85.9	**89.0**	84.0

Tree

Budget	Nodes	Static i.i.d.	Wasserstein	WARP
Small	All	62.9	67.7	**71.2**
	Init	72.7	71.3	**73.0**
	Mid	55.3	63.6	**65.9**
	End	64.3	68.9	**73.4**
Medium	All	**78.9**	77.2	76.1
	Init	85.8	80.7	**86.0**
	Mid	73.3	**75.0**	72.5
	End	**80.0**	77.3	75.4
Large	All	86.7	**90.3**	90.0
	Init	95.3	94.5	**95.4**
	Mid	83.1	**88.2**	86.2
	End	86.4	90.4	**90.5**

Table 3: Accuracy (%, ten-seed mean) on the structured arithmetic-operator benchmarks. INIT/MID/END denote the root, internal nodes, and leaves for the tree, and $L1$, $L2$–$L7$, and $L8$ for the cube.

Setup. We use the 2-layer transformer and compare *static i.i.d.*, *Wasserstein*, and WARP over 10 seeds on the cube and tree benchmarks shown in Figure 6. In both benchmarks, each node is a level in the same local problem family, but the connectivity differs. Appendix B.5 gives the exact graph constructions and endpoint distributions.

Results. Table 3 shows that geometry-aware curricula remain useful beyond one-dimensional orderings, with the clearest gains near the hard end. On the cube, *Wasserstein* consistently improves the hard endpoint $L8$, while *Static i.i.d.* remains strongest on the easier regions and on aggregate at medium and large budgets. On the tree, both geometry-aware curricula are competitive: WARP is strongest in the small-budget regime, especially away from the root, whereas *Wasserstein* becomes strongest overall in the large-budget regime. These experiments show that the same path construction extends to graph-structured difficulty spaces, with effects that vary across graph, region, and training budget.

9 Conclusion

We use Wasserstein interpolation as a controlled lens for studying curriculum learning. This lens lets us vary calibrated difficulty axes, ordering, matched cumulative exposure, endpoint smoothness, pacing, and geometry within one framework. Taken together, our results do not support a single universally best curriculum. Instead, they show that different components of curriculum design matter in different ways: easy-to-hard ordering helps beyond cumulative exposure, pacing and endpoint design change where a fixed budget is most effective, adaptive geometry can yield gains, and structured difficulty spaces make

those gains more localized and more dependent on structure. At the same time, our real-data SFT results suggest that these effects weaken substantially in settings with pretrained models, natural benchmarks, and noisier difficulty labels.

Limitations and Future Work. We intentionally constrain most of the study to from-scratch training and newly constructed tasks, reducing confounding from pretrained knowledge but leaving open how the results transfer to pretrained settings, natural data, and tasks whose difficulty structure interacts with prior knowledge. We also showcase two extensions of the framework, adaptive geometry and structured difficulty spaces, pointing to next directions such as learning geometry or pacing from transfer signals, defining Wasserstein curricula on richer structures, and testing them in pretrained settings.

Disclosure of LLM Use

We used LLM-based assistants during this project for parts of task design, code implementation, exploratory analysis, plotting and analysis code, and drafting or revising portions of the manuscript. These tools were used interactively under close author supervision. The authors made the final research decisions, ran and verified the reported experiments and analyses, checked the resulting figures and tables, and reviewed and edited all final paper text.

Acknowledgements

The majority of this work was done while CS and DAM were at Microsoft Research. CS thanks Brenden Lake for supporting conference travel. DAM additionally acknowledges support from the Kempner Institute for the Study of Natural and Artificial Intelligence, the Aramont Fellowship Fund and NSF Award No. 2229881, NSF AI Institute for Societal Decision Making (NSF AI-SDM).

References

Shun-ichi Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer Japan, 2016. doi: 10.1007/978-4-431-55978-8. URL https://doi.org/10.1007/978-4-431-55978-8.

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 Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32570–32587. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1564. URL https://aclanthology.org/2025.acl-long.1564/.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. Smollm - blazingly fast and remarkably powerful. Hugging Face blog, 2024. URL https://huggingface.co/blog/smollm.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pp. 41–48. ACM, 2009. doi: 10.1145/1553374.1553380.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv.org*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. DOGE: Domain reweighting with generalization estimation. In *International Conference on Machine Learning*, pp. 12895–12915. PMLR, 2024. doi: 10.48550/arXiv.2310.15393. URL https://proceedings.mlr.press/v235/fan24e.html.

Albert Ge, Tzu-Heng Huang, John Cooper, Avi Trost, Ziyi Chu, Satya Sai Srinath Namburi Gnvv, Ziyang Cai, Kendall Park, Nicholas Roberts, and Frederic Sala. R&b: Domain regrouping and data mixture balancing for efficient foundation model training. *arXiv.org*, 2025. doi: 10.48550/arXiv.2505.00358. URL https://arxiv.org/abs/2505.00358.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.21/.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, pp. 1311–1320, 2017. URL https://proceedings.mlr.press/v70/graves17a.html.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pp. 2535–2544, 2019. URL https://proceedings.mlr.press/v97/hacohen19a.html.

Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegreffe. The unreasonable effectiveness of easy training data for hard tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7002–7024, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.378. URL https://aclanthology.org/2024.acl-long.378/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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 35*, pp. 10656–10670. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-0774. URL https://arxiv.org/abs/2210.10195.

Yaning Jia, Chunhui Zhang, Xingjian Diao, Xiangchi Yuan, Z. Ouyang, and S. Vosoughi. What makes a good curriculum? disentangling the effects of data ordering on llm mathematical reasoning. *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34472–34488, 2026. doi: 10.18653/v1/2026.acl-long.1591. URL https://arxiv.org/abs/2510.19099.

Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11341–11358. PMLR, 2022. URL https://proceedings.mlr.press/v162/klink22a.html.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Neural Information Processing Systems*, 2010. URL https://proceedings.neurips.cc/paper/2010/hash/e57c6b956a6521b28495f2886ca0977a-Abstract.html.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yong 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. In *Advances in Neural Information Processing Systems 38*, pp. 111997–112018. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025a. doi: 10.52202/085713-3380. URL https://openreview.net/forum?id=XtNiw8OQsy. Poster.

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. In *International Conference on Learning Representations*, 2025b. doi: 10.48550/arXiv.2407.01492. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5f67d864aae6115374fed7beddd119e0-Abstract-Conference.html.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, 2020. doi: 10.1109/TNNLS.2019.2934906. URL https://doi.org/10.1109/TNNLS.2019.2934906.

Robert J. McCann. A convexity principle for interacting gases. *Advances in Mathematics*, 128 (1):153–179, 1997. doi: 10.1006/aima.1997.1634.

Guangyu Meng, Qinkai Zeng, John P. Lalor, and Hong Yu. A psychology-based unified dynamic framework for curriculum learning. *Computational Linguistics*, pp. 1–49, 2026. doi: 10.1162/COLI.a.584. URL https://direct.mit.edu/coli/article/doi/10.1162/COLI.a.584/134522/A-Psychology-based-Unified-Dynamic-Framework-for.

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*, 2025. doi: 10.48550/arXiv.2502.16802. URL https://arxiv.org/abs/2502.16802.

G. PeyrÃ© and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607, 2019. doi: 10.1561/2200000073.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, B. PÃ³czos, and Tom Michael Mitchell. Competence-based curriculum learning for neural machine translation. In *North American Chapter of the Association for Computational Linguistics*, pp. 1162–1172. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1119. URL https://aclanthology.org/N19-1119/.

Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Birkhäuser Cham, 2015. doi: 10.1007/978-3-319-20828-2.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Computer Vision and Pattern Recognition*, pp. 761–769. IEEE, 2016. doi: 10.1109/CVPR.2016.89.

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 *Conference on Empirical Methods in Natural Language Processing*, pp. 9275–9293. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746/.

Mariya Toneva, Alessandro Sordoni, RÃ©mi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

CÃ©dric Villani. *Optimal transport, Old and New*, volume 338. Springer Science & Business Media, 2008. ISBN 9783540710493. doi: 10.1007/978-3-540-71050-9.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2022. doi: 10.1109/TPAMI.2021.3069908. URL https://doi.org/10.1109/TPAMI.2021.3069908.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tW4QEInpni.

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. In *Advances in Neural Information Processing Systems 36*, pp. 69798–69818. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-3059. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2403.16952. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cc84bfabe6389d8883fc2071c848f62a-Abstract-Conference.html.

A Extended Related Work

Curriculum learning and difficulty-aware training. Classical curriculum learning moves from easy to hard and can stabilize optimization and improve generalization (Bengio et al., 2009). Automated curriculum-learning methods choose what data to present and when: self-paced learning matches example difficulty to learner competence (Kumar et al., 2010); teacher–student curricula prioritize items with high learning progress (Matiisen et al., 2020); bandit-style schedulers adapt task proportions online (Graves et al., 2017); and competence-based schedules control how quickly harder examples are introduced (Platanios et al., 2019). Evidence suggests that pacing strongly affects outcomes (Hacohen & Weinshall, 2019), while training-dynamics diagnostics provide signals for difficulty and ambiguity (Swayamdipta et al., 2020); related heuristics include hard-example mining and forgetting-based selection (Shrivastava et al., 2016; Toneva et al., 2019). Our framework instead represents a curriculum as a path over distributions on discrete difficulty levels and separates the path itself from its pacing. This makes it possible to move probability mass smoothly through nearby levels while controlling where the curriculum travels and how quickly it advances.

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 (McCann, 1997; PeyrÃ© & Cuturi, 2019; Santambrogio, 2015). OT has also been used to *construct curricula* in reinforcement learning by generating geodesic sequences or constrained OT plans between task distributions (Huang et al., 2022; Klink et al., 2022). Our approach is related in that it also uses Wasserstein geodesics for curriculum learning, but the emphasis here is empirical control rather than a new RL-specific scheduler. We use Wasserstein paths as a common scaffold for comparing endpoint choice, smoothness, traversal speed, and adaptive geometry on calibrated synthetic tasks and language-model reasoning benchmarks. This keeps the distributional formulation fixed while studying which curriculum-design choices matter.

Data Mixing. Designing training distributions from heterogeneous sources is an important part of large-scale model training. Static approaches fix mixture weights in advance, e.g., DoReMi (Xie et al., 2023) optimizes domain sampling via a proxy model, and DoGE (Fan et al., 2024) estimates domain importance from small runs. Recent work also predicts or balances effective mixtures from proxy runs and learned signals: RegMix regresses performance from proxy runs (Liu et al., 2025b), data mixing laws model weight–loss functions (Ye et al., 2025), mixtures of data experts approximate source contributions (Belenki et al., 2025), and regroup-and-balance strategies adapt mixtures during training (Ge et al., 2025). Other work goes beyond source-level mixing, such as topic-based mixtures (Peng et al., 2025). 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.

B Detailed Experimental Setup

We include detailed experiment setups and data examples.

B.1 Task and Dataset Details

Each difficulty axis below is described using the writing terms from our task-to-difficulty-axis mapping. When one difficulty axis varies, the remaining task parameters stay at the dataset defaults stated in the same paragraph.

B.1.1 k-Parity Dataset

Task. Each example is a binary string $x \in \{0,1\}^n$ with label

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

Only the leading bits matter; the remaining bits are nuisance features. We report classification accuracy.

Construction and difficulty levels. We study two difficulty axes, each with 5 levels:

- **Informative-bit difficulty axis:** levels use $\{1,2,3,4,5\}$ informative prefix bits while keeping the total string length fixed at 32.
- **Length difficulty axis:** levels use total string lengths $\{24,28,32,36,40\}$ while keeping the number of informative prefix bits fixed at 3.

For the informative-bit difficulty axis, later levels strictly contain earlier ones: before the final level, the later informative positions are fixed to 0 and are released only when that level is introduced. We therefore also evaluate each level on a unique slice where the newly introduced informative bit is 1 and all later informative positions remain 0, so that each test set isolates the new bit introduced at that level.

Example.

Input. `01100110001000100001010101010110`
Target output. `0`

In this string, the first three bits are $0,1,1$, so the correct label is $0 \oplus 1 \oplus 1 = 0$. A model that still behaves like the previous level, using only the first two bits, would predict $0 \oplus 1 = 1$ instead. This is exactly the failure that the unique-slice evaluation is designed to expose.

Dataset size. We use batch size 1000. For the informative-bit difficulty axis, the small, medium, and large budgets are 100, 200, and 300 steps. For the length difficulty axis, they are 5000, 7500, and 10000 steps.

B.1.2 Dyck Language Dataset

Task. Each example is a prefix-completion problem for the Dyck language. We sample a valid bracket string, reveal an initial prefix, and ask the model to output the full completed sequence, not just the missing tail. We report exact-match accuracy on the completed string.

Construction and difficulty levels. We vary one factor at a time:

- **Bracket-type difficulty axis:** 4 levels with $\{1,2,3,4\}$ bracket types, while total sequence length stays fixed at 20 and hidden suffix length stays fixed at 3.

- **Length difficulty axis:** 4 levels with total sequence lengths $\{14, 16, 18, 20\}$, while the number of bracket types stays fixed at 4 and the hidden suffix length stays fixed at 3.
- **Missing-suffix difficulty axis:** 4 levels with hidden suffix lengths $\{3, 5, 7, 9\}$, while the number of bracket types stays fixed at 4 and the total sequence length stays fixed at 20.

Example.

> *Input prefix.* `[[(){}]][]()()[<<`
> *Target output.* `[[(){}]][]()()[<<>>]`

The missing part is short, but the model still has to track the open brackets that remain unresolved and finish the entire balanced string correctly.

Dataset size. We use batch size 100. For the bracket-type difficulty axis, the small, medium, and large budgets are 200, 350, and 1000 steps; for the length difficulty axis they are 200, 300, and 1000; and for the missing-suffix difficulty axis they are 300, 500, and 1000.

B.1.3 Survo Dataset

Task. Survo asks the model to fill missing entries in a partially observed 4×4 grid whose last row and last column give row and column sums. The unknown cells always lie in the upper-left 3×3 block, and we report exact-match accuracy on the completed grid.

Construction and difficulty levels. We use two difficulty axes:

- **Blank-count difficulty axis:** 5 levels with $\{1, 2, 3, 4, 5\}$ blanks, while the grid size stays fixed at 4×4 and the allowed cell range stays fixed to 1–9.
- **Value-range difficulty axis:** 4 levels with maximum allowed cell values $\{6, 9, 12, 15\}$, while the grid size stays fixed at 4×4, the number of blanks stays fixed at 3, and the minimum cell value stays fixed at 1.

Example.

> *Input.* `[[X, 1, 2, 9], [6, X, 1, 10], [4, 2, X, 8], [16, 6, 5, 27]]`
> *Target output.* `[[6, 1, 2, 9], [6, 3, 1, 10], [4, 2, 2, 8], [16, 6, 5, 27]]`

The first row must begin with 6 because $6 + 1 + 2 = 9$; the second row must place 3 in the middle because $6 + 3 + 1 = 10$; and the third row then takes 2.

Dataset size. We use batch size 100. For both difficulty axes, the small, medium, and large budgets are 7500, 15000, and 25000 steps.

B.1.4 Dyck Language Errors Dataset

Task. This task presents a bracket string and asks for the first position at which it becomes invalid, using 1-indexing. Errors include an unmatched closing bracket, a closing bracket of the wrong type, or a prefix that is locally valid but still unfinished at the end; in the last case the answer is one position past the end of the string. Valid strings receive -1. We report exact-match accuracy.

Construction and difficulty levels. We use two difficulty axes, each with 4 levels:

- **Bracket-type difficulty axis:** levels use $\{1, 2, 3, 4\}$ bracket types while keeping the total string length fixed at 20.
- **Length difficulty axis:** levels use total string lengths $\{10, 15, 20, 25\}$ while keeping the number of bracket types fixed at 3.

Example.

Input. `[]((()))([()][[]((()(`
Target output. `7`

The first six symbols can still be parsed consistently, but the seventh symbol is an extra closing parenthesis with nothing left to match.

Dataset size. We use batch size 100. For the bracket-type difficulty axis, we use 8,000 training examples and 1,000 evaluation examples per level, and the small, medium, and large budgets are 300, 600, and 1,000 steps. For the length difficulty axis, we keep the original setup with 2,400 training examples and 2,000 evaluation examples per level, and the small, medium, and large budgets are 750, 1500, and 3000.

B.1.5 Calendar Scheduling Dataset

Task. Each example specifies task durations, precedence constraints, and a finite time horizon for a single-machine schedule with no overlap. The solver always picks the alphabetically earliest available task and starts it immediately after the previous task ends. The model must output the full schedule as `Task:start-end` pairs, and we report exact-match accuracy on that string.

Construction and difficulty levels. We vary three factors separately:

- **Task-count difficulty axis:** 4 levels with $\{4, 6, 8, 10\}$ tasks, while the time horizon stays fixed at 20, the number of precedence constraints stays fixed at 4, and the maximum task duration stays fixed at 3.
- **Horizon difficulty axis:** 4 levels with time horizons $\{12, 16, 20, 24\}$, while the number of tasks stays fixed at 6, the number of precedence constraints stays fixed at 4, and the maximum task duration stays fixed at 3.
- **Dependency difficulty axis:** 4 levels with $\{2, 4, 6, 8\}$ precedence constraints, while the number of tasks stays fixed at 6, the time horizon stays fixed at 20, and the maximum task duration stays fixed at 3.

Example.

Input. `Time horizon 0--19; durations A=1 B=1 C=1 D=2 E=2 F=1;`
`dependencies B->D, B->E, A->D, E->F`
Target output. `A:0-1 B:1-2 C:2-3 D:3-5 E:5-7 F:7-8`

Tasks A, B, and C are initially available, so the alphabetical rule places them first. Task D must wait for both A and B, and task F must wait for E.

Dataset size. We use batch size 100. For the task-count difficulty axis, the small, medium, and large budgets are 2000, 3000, and 4500 steps; for the horizon difficulty axis they are 1400, 1700, and 3000; and for the dependency difficulty axis they are 2000, 3000, and 4200.

B.1.6 Game of 24 Dataset

Task. Each instance gives four integers and a target value. The model must produce an arithmetic expression that uses each number exactly once and combines them with the standard binary operations to reach the target. Because many expressions can be correct, we report verifier-based success rate rather than exact string match.

Construction and difficulty levels. We use two difficulty axes, each with 5 levels:

- **Target-value difficulty axis:** levels use target values $\{18, 21, 24, 27, 30\}$, while the number of input numbers stays fixed at four, the allowed operations stay fixed to $+, -, \times, \div$, and the input-number range stays fixed at 1–15.

- **Number-range difficulty axis:** levels use input-number ranges $\{1\text{–}12, 1\text{–}15, 1\text{–}18, 1\text{–}21, 1\text{–}24\}$, while the target value stays fixed at 24 and the number of input numbers stays fixed at four.

The second change creates a broader and less repetitive arithmetic search space.

Example.

Input. `Numbers: 3, 11, 4, 8; target: 24`
Target output. `4*((3+11)-8)`

Any verifier-equivalent expression is counted as correct; this is one valid target output.

Dataset size. We use batch size 100. For the target-value difficulty axis, the small, medium, and large budgets are 4200, 5000, and 9500 steps. For the number-range difficulty axis, they are 2500, 5000, and 9000.

B.1.7 Goods Exchange Dataset

Task. Each example starts from a one-to-one ownership map between people and objects, followed by a short sequence of exchange statements. The model must return the final ownership map, sorted alphabetically and formatted one line per person as `Person: item`. We report exact-match accuracy after canonicalizing outputs to that format.

Construction and difficulty levels. We use two difficulty axes, each with 4 levels:

- **People-count difficulty axis:** levels use $\{3, 4, 5, 6\}$ people, and therefore the same number of items, while keeping the number of exchange statements fixed at 2.
- **Exchange-length difficulty axis:** levels use $\{2, 3, 4, 5\}$ exchange statements while keeping the number of people fixed at 3.

The first broadens the state being tracked; the second lengthens the chain of ownership updates.

Example.

Input. `Initial ownership: Sophia -> coral kettle; Betty -> beige kettle; Donna -> beige watch.`
`Statements: Betty asked to exchange with Donna, but Donna refused; Sophia and Donna exchanged items; Sophia and Betty exchanged items.`
Target output. `Betty: beige watch`
`Donna: coral kettle`
`Sophia: beige kettle`

Here the first statement is a decoy: Donna refuses Betty's request, so ownership does not change. The next two exchanges do the real work. After Sophia and Donna exchange items, Donna holds the coral kettle and Sophia holds the beige watch; after Sophia and Betty exchange items, Betty ends with the beige watch and Sophia with the beige kettle.

Dataset size. We use batch size 100. For the people-count difficulty axis, the small, medium, and large budgets are 500, 1000, and 1500 steps. For the exchange-length difficulty axis, they are 1500, 2250, and 3000.

B.1.8 Index Select Dataset

Task. The input is a token sequence together with a list of inclusive 0-indexed ranges. The model must concatenate the selected spans in order and output the resulting token list, space-separated. We report exact-match accuracy.

Construction and difficulty levels. We vary four factors separately:

- **Source-length difficulty axis:** 5 levels with source sequence lengths $\{10, 12, 14, 16, 18\}$, while selected-output length stays fixed at 8, the number of ranges stays fixed at 3, and the alphabet size stays fixed at 4.
- **Selected-output-length difficulty axis:** 5 levels with $\{4, 6, 8, 10, 12\}$ selected tokens, while the source sequence length stays fixed at 16, the number of ranges stays fixed at 3, and the alphabet size stays fixed at 4.
- **Range-count difficulty axis:** 5 levels with $\{1, 2, 3, 4, 5\}$ ranges, while the source sequence length stays fixed at 16, the selected-output length stays fixed at 8, and the alphabet size stays fixed at 4.
- **Alphabet-size difficulty axis:** 5 levels with alphabet sizes $\{2, 3, 4, 5, 6\}$, while the source sequence length stays fixed at 16, the selected-output length stays fixed at 8, and the number of ranges stays fixed at 3.

These changes make the copy operation longer, more fragmented, or less repetitive.

Example.

> *Input.* `Sequence: d a d a a b a b a d d a b c a c; ranges: 0-1, 3-5, 10-12`
> *Target output.* `d a a a b d a b`

The range `0-1` contributes `d a`, the range `3-5` contributes `a a b`, and the range `10-12` contributes `d a b`.

Dataset size. We use batch size 100. For source length and selected-output length, the small, medium, and large budgets are 3000, 4000, and 6000 steps. For the number of ranges they are 3600, 4000, and 6400, and for alphabet size they are 2500, 3400, and 6000.

B.1.9 Sequence Reverse Dataset

Task. This is the standard sequence-reversal task: given a token sequence, output the same sequence in reverse order. We report exact-match accuracy on the reversed sequence.

Construction and difficulty levels. We use two difficulty axes, each with 5 levels:

- **Length difficulty axis:** levels use sequence lengths $\{6, 8, 10, 12, 14\}$ while keeping the alphabet size fixed at 4.
- **Alphabet-size difficulty axis:** levels use alphabet sizes $\{2, 3, 4, 5, 6\}$ while keeping the sequence length fixed at 12.

Example.

> *Input.* `d b c b b c b b b c a a`
> *Target output.* `a a c b b b c b b c b d`

Dataset size. We use batch size 100. For both difficulty axes, the small, medium, and large budgets are 600, 700, and 800 steps.

B.1.10 Run-Length Encoding Dataset

Task. Each example contains a token sequence that must be converted to run-length form as space-separated `<count> <symbol>` pairs. The task is deterministic, and we report exact-match accuracy on the encoded string.

Construction and difficulty levels. We vary four properties separately:

- **Length difficulty axis:** 5 levels with sequence lengths $\{6, 7, 8, 9, 10\}$, while the maximum number of runs stays fixed at 8, the alphabet size stays fixed at 4, and the maximum run length stays fixed at 4.

- **Run-count difficulty axis:** 5 levels with maximum numbers of runs $\{4, 5, 6, 7, 8\}$, while the sequence length stays fixed at 12, the alphabet size stays fixed at 4, and the maximum run length stays fixed at 4.
- **Alphabet-size difficulty axis:** 4 levels with alphabet sizes $\{2, 3, 4, 5\}$, while the sequence length stays fixed at 12, the maximum number of runs stays fixed at 8, and the maximum run length stays fixed at 4.
- **Run-length difficulty axis:** 4 levels with maximum run lengths $\{2, 3, 4, 5\}$, while the sequence length stays fixed at 12, the maximum number of runs stays fixed at 8, and the alphabet size stays fixed at 4.

These difficulty axes make the compressed description longer, more segmented, or more varied.

Example.

Input. `c a a a d d d d b b b b`
Target output. `1 c 3 a 4 d 4 b`

The sequence contains one `c`, then three `a`'s, then four `d`'s, and finally four `b`'s.

Dataset size. We use batch size 100. The small, medium, and large budgets are 1000, 1500, and 2200 steps when varying sequence length; 1000, 1350, and 2100 when varying the number of runs; 1000, 1500, and 2100 when varying alphabet size; and 1300, 1500, and 2200 when varying maximum run length.

B.1.11 Stack Operations Dataset

Task. This task simulates a LIFO stack over a finite alphabet. Inputs are sequences of `PUSH` and `POP` operations, and the model must output the final stack from top to bottom, or `EMPTY` if no items remain. We report exact-match accuracy.

Construction and difficulty levels. We vary four factors separately:

- **Operation-count difficulty axis:** 4 levels with $\{6, 8, 10, 12\}$ operations, while the alphabet size stays fixed at 4 and the pop probability stays fixed at 0.4; for this difficulty axis, the stack-depth cap scales with the operation count, with a minimum cap of 4.
- **Alphabet-size difficulty axis:** 4 levels with alphabet sizes $\{2, 3, 4, 5\}$, while the number of operations stays fixed at 8, the maximum stack depth stays fixed at 6, and the pop probability stays fixed at 0.4.
- **Maximum-depth difficulty axis:** 4 levels with maximum stack depths $\{2, 3, 4, 5\}$, while the number of operations stays fixed at 8, the alphabet size stays fixed at 4, and the pop probability stays fixed at 0.4.
- **Pop-rate difficulty axis:** 4 levels with pop probabilities $\{0.65, 0.5, 0.35, 0.2\}$, listed from easy to hard, while the number of operations stays fixed at 8, the alphabet size stays fixed at 4, and the maximum stack depth stays fixed at 6.

Later levels therefore produce longer traces and, in the last difficulty axis, more live stack contents to remember.

Example.

Input. `PUSH a; PUSH a; POP; POP; PUSH d; POP; PUSH d; PUSH d`
Target output. `d d`

The first two pushes are canceled by the next two pops, the next `PUSH d` is immediately removed, and the last two pushes remain.

Dataset size. We use batch size 100. The small, medium, and large budgets are 300, 380, and 550 steps when varying the number of operations; 280, 420, and 600 when varying alphabet size; 180, 220, and 280 when varying maximum depth; and 280, 360, and 540 when varying the pop rate.

B.1.12 Queue Operations Dataset

Task. This task simulates a FIFO queue. Inputs are `ENQUEUE` and `DEQUEUE` operations, and the model must output the final queue from front to back, or `EMPTY` if the queue is empty. We report exact-match accuracy.

Construction and difficulty levels. We report three difficulty axes for this task:

- **Operation-count difficulty axis:** 5 levels with $\{12, 16, 20, 24, 28\}$ operations while keeping the other task parameters fixed at their default values.
- **Alphabet-size difficulty axis:** 4 levels with alphabet sizes $\{4, 6, 8, 10\}$ while keeping the remaining parameters fixed.
- **Queue-capacity difficulty axis:** 4 levels with capacities $\{2, 3, 4, 5\}$, listed from easy to hard, while keeping the remaining parameters fixed.

Example.

> *Input.* `ENQUEUE b; ENQUEUE d; DEQUEUE; DEQUEUE; ENQUEUE b; DEQUEUE; ENQUEUE c; ENQUEUE a; ENQUEUE d; ENQUEUE d; DEQUEUE; DEQUEUE`
> *Target output.* `d d`

The first three dequeues remove `b`, then `d`, then the later `b`. The remaining front-to-back queue is `d d`.

Dataset size. We use batch size 100. The small, medium, and large budgets are 450, 570, and 720 steps when varying the number of operations; 300, 400, and 640 when varying alphabet size; and 280, 390, and 550 when varying the queue-cap parameter.

B.2 Architecture and Training Details

All models were trained on a single NVIDIA A100 GPU.

Scratch transformer tasks Most tasks in the current synthetic sweep, including k-Parity, Dyck Language, Dyck Language Errors, Sequence Reverse, Index Select, Run-Length Encoding, Stack Operations, Queue Operations, Survo, and Calendar Scheduling, are trained from scratch as decoder-only transformers. Most use hidden size 256, 4 layers, and 8 attention heads with AdamW, learning rate 5×10^{-5}, weight decay 0.1, gradient clipping at 1.0, and a constant learning-rate schedule with no warmup. The main architectural exceptions are k-Parity, which uses a smaller hidden size 128 / 1-layer / 4-head model and learning rate 5×10^{-4} with batch size 1,000; Dyck Language, which uses 3 layers; Survo, which uses 6 layers; and Calendar Scheduling, which uses a larger 6-layer model with hidden size 384 and 12 attention heads. Other scratch tasks typically use train batch size 100, with task-specific evaluation batch sizes and calibrated small/medium/large budgets given in the corresponding dataset subsections and task defaults.

Pretrained language-like tasks. GAME OF 24 and GOODS EXCHANGE use `HuggingFaceTB/SmolLM-135M` (Ben Allal et al., 2024). We fine-tune these models with AdamW using learning rate 5×10^{-5}, weight decay 0.1, gradient clipping at 1.0, and a constant learning-rate schedule without warmup.

B.3 Level-Wise Cumulative Exposure

Figure 7 shows the cumulative level-wise exposure induced over the full training trajectory with $\gamma = 1$ and $\tau = 1$. Uniform sampling gives equal exposure to all levels, linear interpolation allocates more exposure to the boundary levels, and Wasserstein interpolation allocates more exposure to the intermediate levels. In the default one-dimensional setting, this Wasserstein path uses the path metric $d(\ell, \ell') = |\ell - \ell'|$, i.e., the standard W_1 geometry on ordered levels. This illustrates that even when different curricula converge to similar

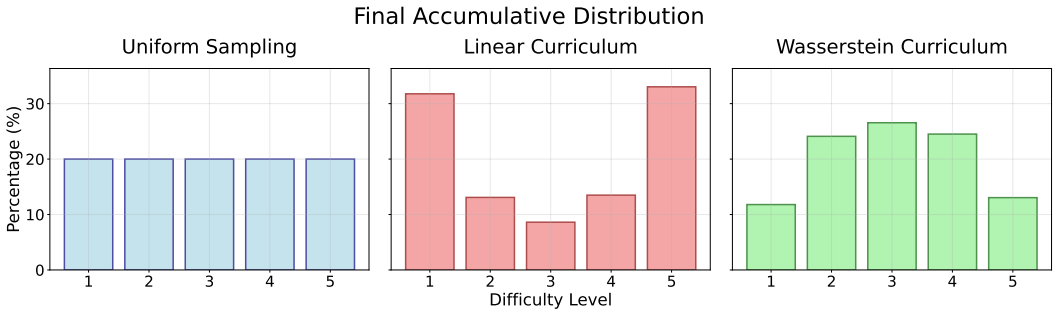


Figure 7: Cumulative level-wise exposure under different curricula. **Left:** uniform sampling; **middle:** linear interpolation; **right:** Wasserstein interpolation.

overall performance, the Wasserstein curriculum achieves better sample efficiency at the higher levels.

B.4 Adaptive Geometry in WARP

Section 7 uses a simple adaptive geometry on an ordered set of difficulty levels. This can be viewed as a discrete, gradient-informed geometry on the level axis. In the default setting, adjacent levels are equally spaced; WARP instead learns nonuniform edge lengths from local gradient alignment, yielding a discrete analogue of a one-dimensional Riemannian metric. This is loosely related to information geometry, where gradient information is used to define local geometry through the Fisher information matrix (Amari, 2016). Here, we use cosine similarity between neighboring level gradients as a simpler measure of local similarity. Let $P_0, P_1 \in \Delta^{m-1}$ denote the easy and hard endpoint distributions, let T be the total number of training steps, and let $T_w = \lfloor \rho T \rfloor$ be a short warmup, with $\rho = 0.05$ in our experiments.

During warmup, WARP uses uniform sampling:

$$P^k_{\text{WARP}} = \text{Unif}(\{1, \ldots, m\}), \qquad k < T_w.$$

At the end of warmup, it probes each level to obtain gradient vectors $g_1, \ldots, g_m$ and forms the cosine-coupling matrix

$$C_{ij} = \frac{\langle g_i, g_j \rangle}{\|g_i\| \, \|g_j\|}.$$

It then converts adjacent couplings into edge lengths

$$\Delta_\ell = 2 - \text{clip}(C_{\ell,\ell+1}, 0, 1), \qquad \ell = 1, \ldots, m-1,$$

defines locations $x_1 = 0$ and $x_{\ell+1} = x_\ell + \Delta_\ell$, and follows the same forward Wasserstein geodesic on the learned locations $x = (x_1, \ldots, x_m)$:

$$\tau_k = \left(\frac{k - T_w}{T - T_w - 1} \right)^\gamma, \qquad P^k_{\text{WARP}} = \mathcal{G}_x(\tau_k; P_0, P_1), \qquad k \geq T_w,$$

where $\mathcal{G}_x$ denotes Wasserstein displacement interpolation on locations x. Thus WARP changes geometry, not endpoints or direction; pacing changes as a consequence. For visualization, Figure 5 rescales the learned locations affinely to a fixed display span; this does not change the relative geometry or the induced pacing pattern.

B.5 Details for Structured Difficulty Spaces

Section 8 uses two structured variants of the same arithmetic-operator family: an 8-node cube and a 7-node tree. Figure 6 visualizes these two benchmark structures. In the cube, each node adds a subset of three operator modifications—multiplication, parentheses, and negation—so the graph follows the natural Boolean-cube structure from the simplest add/sub benchmark to the all-three endpoint. In the tree, the root splits first into multiplication and parentheses branches, and each branch then refines into longer-expression and

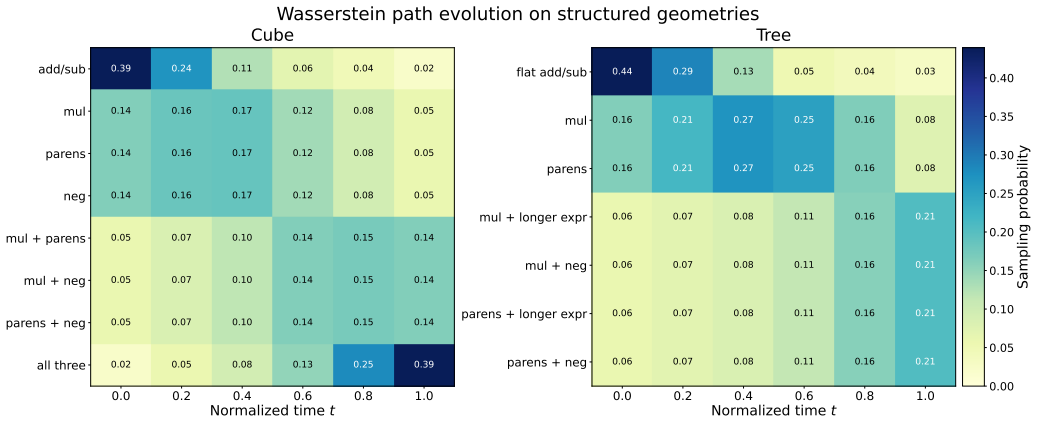


Figure 8: Wasserstein path evolution on the cube and tree structures used in Section 8. Each column corresponds to normalized training progress $t \in \{0.0, 0.2, \ldots, 1.0\}$, and each cell shows the sampling probability assigned to that node along the graph-Wasserstein path.

negation leaves. In both cases, nodes are grouped into *easy*, *mid*, and *hard* regions; for the cube these correspond to $L1$, $L2$–$L7$, and $L8$, while for the tree they correspond to the root, internal branch nodes, and leaves.

For the cube, the Wasserstein endpoints over $L1$–$L8$ are $(0.39, 0.14, 0.14, 0.14, 0.05, 0.05, 0.05, 0.02)$ and $(0.02, 0.05, 0.05, 0.05, 0.14, 0.14, 0.14, 0.39)$. For the tree, the endpoints over $L1$–$L7$ are $(0.44, 0.16, 0.16, 0.06, 0.06, 0.06, 0.06)$ and $(0.03, 0.08, 0.08, 0.21, 0.21, 0.21, 0.21)$. The main comparison on structured difficulty spaces reports ten seeds for *static i.i.d.*, *Wasserstein*, and WARP.

C Detailed Results and Mechanism Analyses

C.1 Task-Wise Final-Accuracy Barplots

Setup. To complement Section 4.1, we report final overall accuracy separately for each task and difficulty axis. Each figure corresponds to one task. Within a figure, panels correspond to the task's difficulty axes, the x-axis groups the small, medium, and large budgets, and colored bars compare the eight curricula: *static i.i.d.*, linear, Wasserstein, WARP, the reverse variants, and the two static-matching baselines. Bars show the mean over 10 random-seed repetitions, and error bars show one standard deviation.

Findings. These plots expose the heterogeneity behind Table 1: the ranking shifts across tasks, difficulty axes, and budgets, rather than collapsing to a single dominant curriculum.

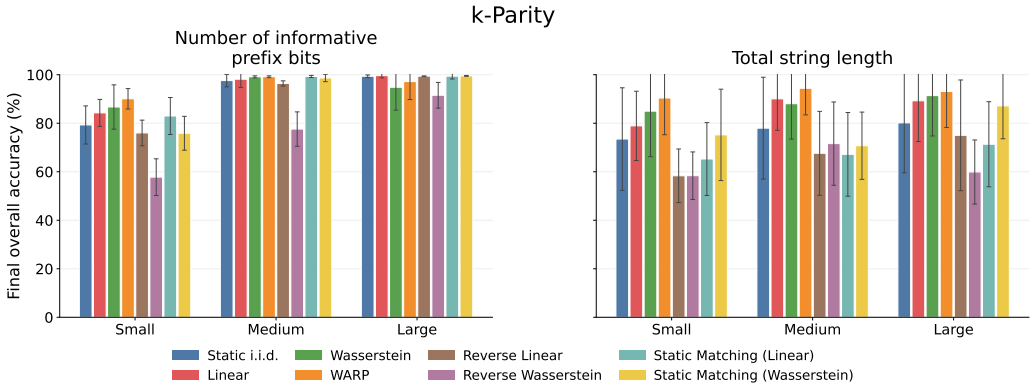


Figure 9: Task-wise final overall accuracy for K-Parity. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

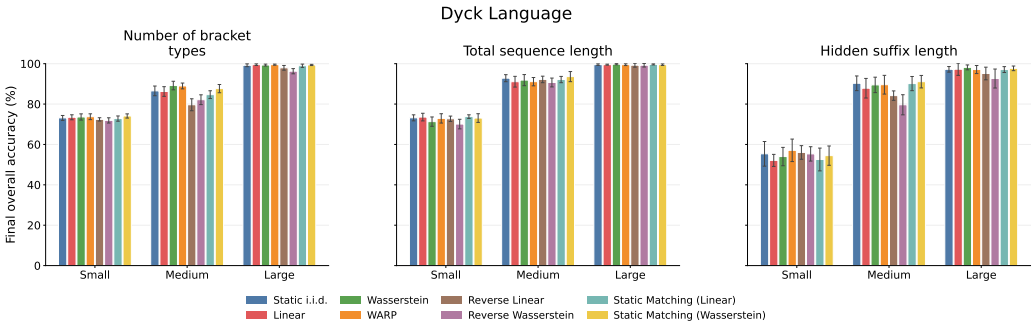


Figure 10: Task-wise final overall accuracy for Dyck Language. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

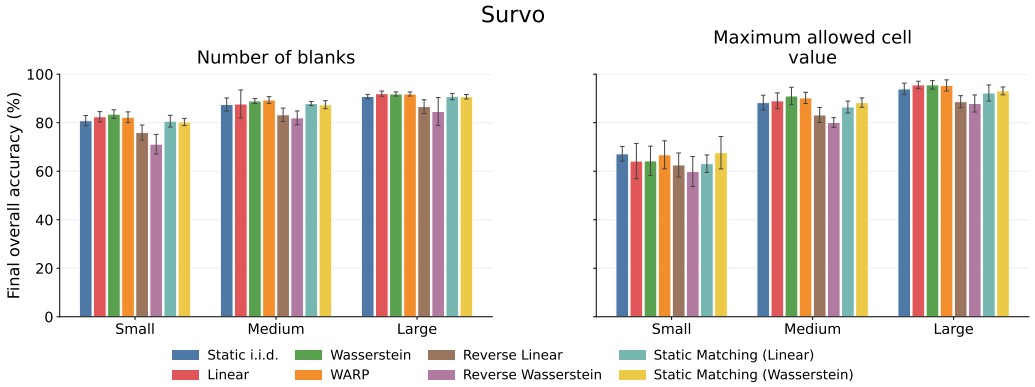


Figure 11: Task-wise final overall accuracy for Survo. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

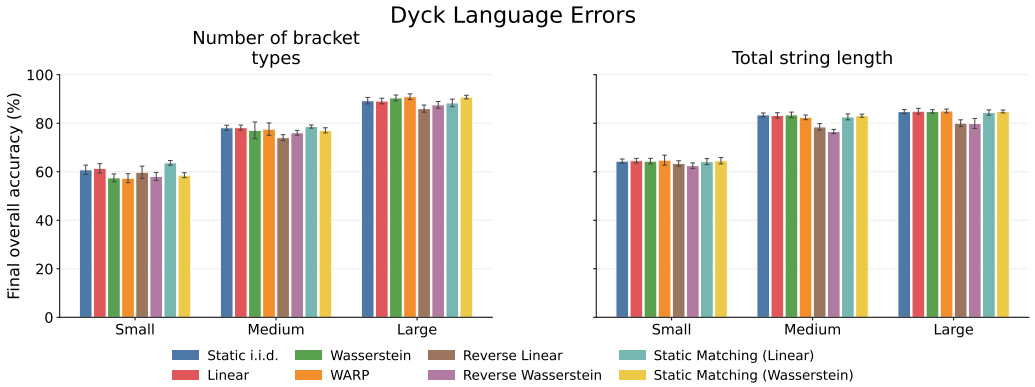


Figure 12: Task-wise final overall accuracy for Dyck Language Errors. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

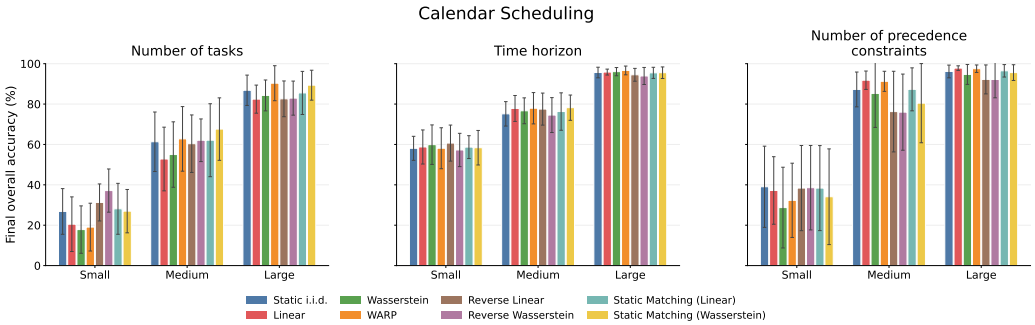


Figure 13: Task-wise final overall accuracy for Calendar Scheduling. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

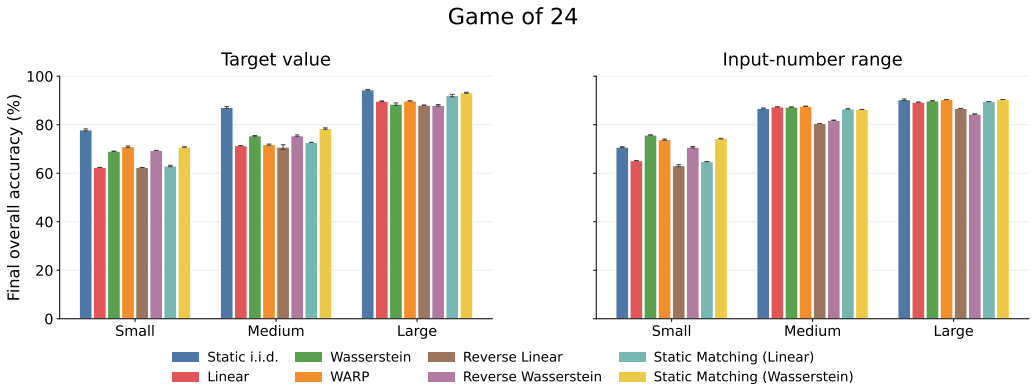


Figure 14: Task-wise final overall accuracy for Game of 24. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

Figure 15: Task-wise final overall accuracy for Goods Exchange. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

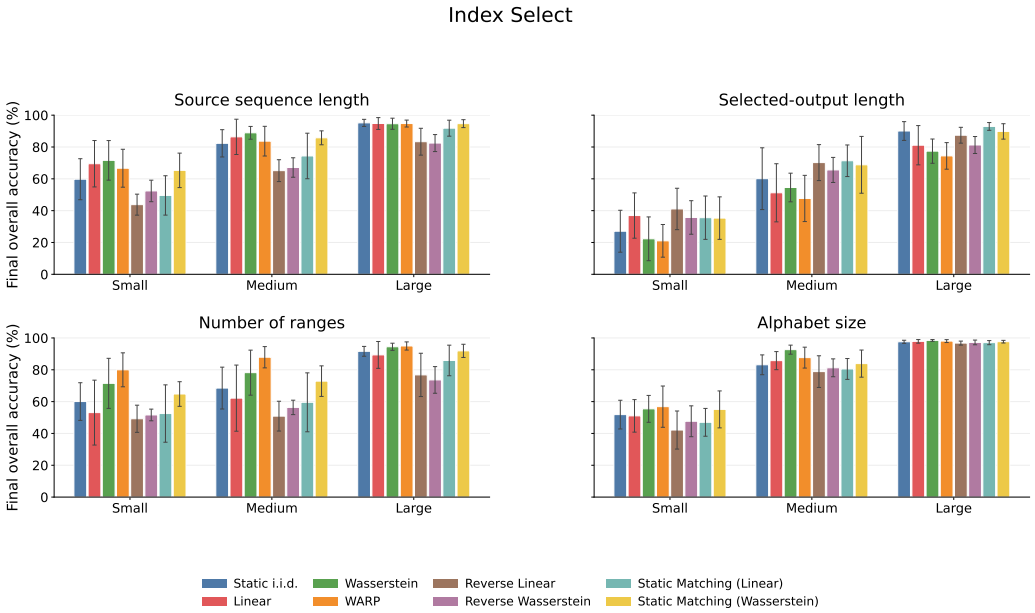


Figure 16: Task-wise final overall accuracy for Index Select. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

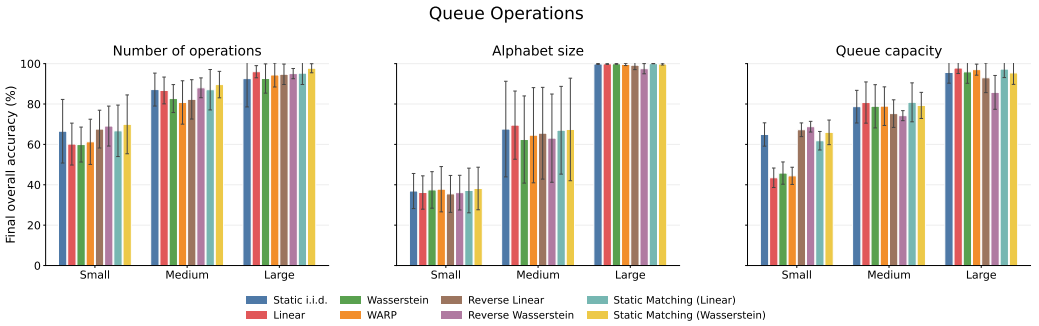


Figure 17: Task-wise final overall accuracy for Queue Operations. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

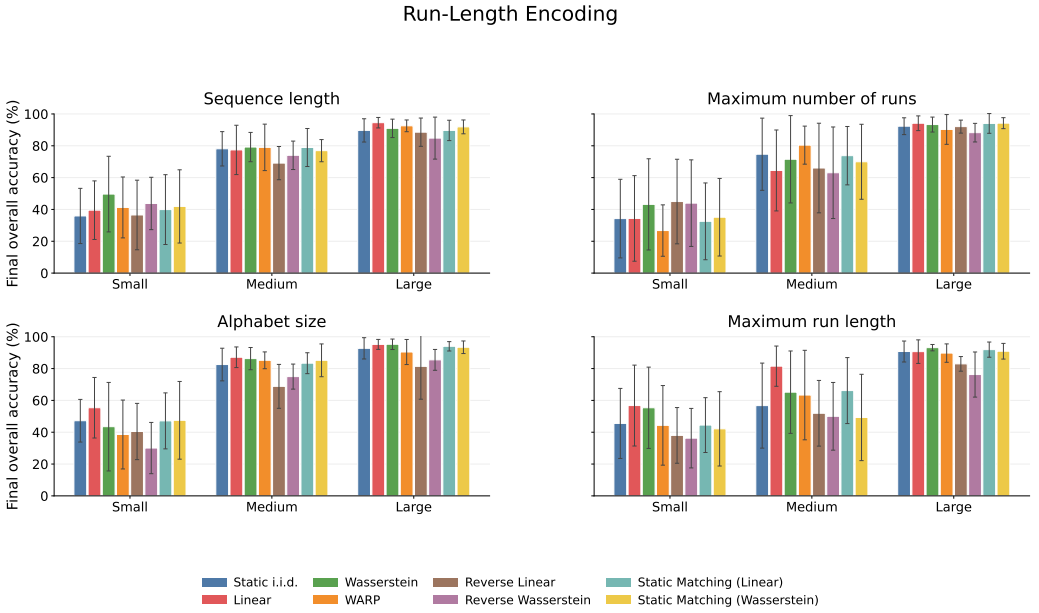


Figure 18: Task-wise final overall accuracy for Run-Length Encoding. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

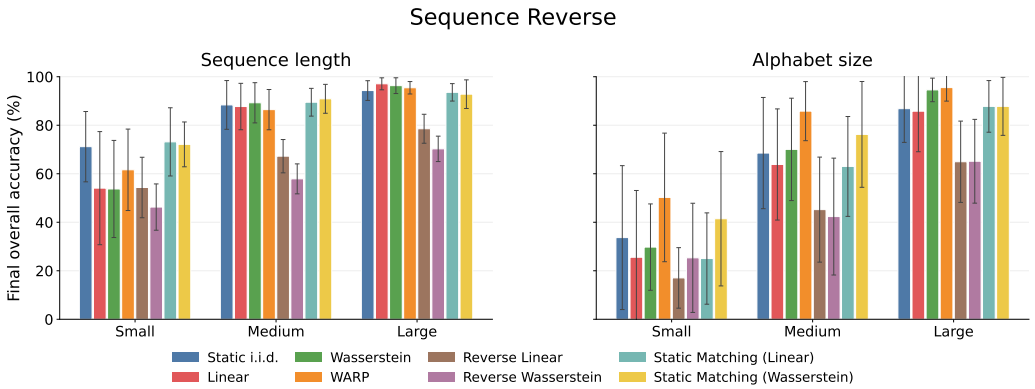


Figure 19: Task-wise final overall accuracy for Sequence Reverse. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

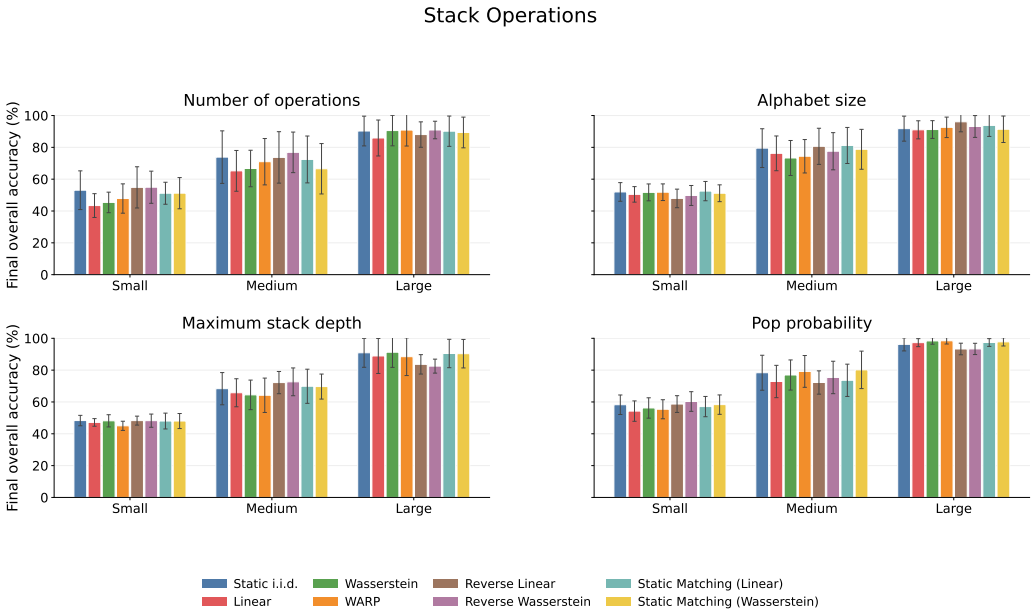


Figure 20: Task-wise final overall accuracy for Stack Operations. Panel titles denote the difficulty axes. Bars show mean final overall accuracy over 10 random-seed repetitions, and error bars show one standard deviation.

C.2 Budget-Wise Exposure, Accuracy, and Exposure-Adjusted Accuracy

Setup. To complement Section 4.2, we report the same easiest, middle, and hardest probe-bucket summaries separately for the small, medium, and large budgets. Exposure is the cumulative share of training mass assigned to each bucket over the full run. As a simple proxy for data efficiency, we use exposure-adjusted accuracy: final bucket accuracy divided by cumulative exposure to that bucket.

Findings. The medium row reproduces Figure 2, while the small and large rows show how the same tradeoff evolves with budget. The qualitative pattern is stable: *static i.i.d.* remains strongest on easy-end accuracy, linear is strongest in exposure-adjusted accuracy in the middle bucket, and Wasserstein is strongest on the hardest bucket.

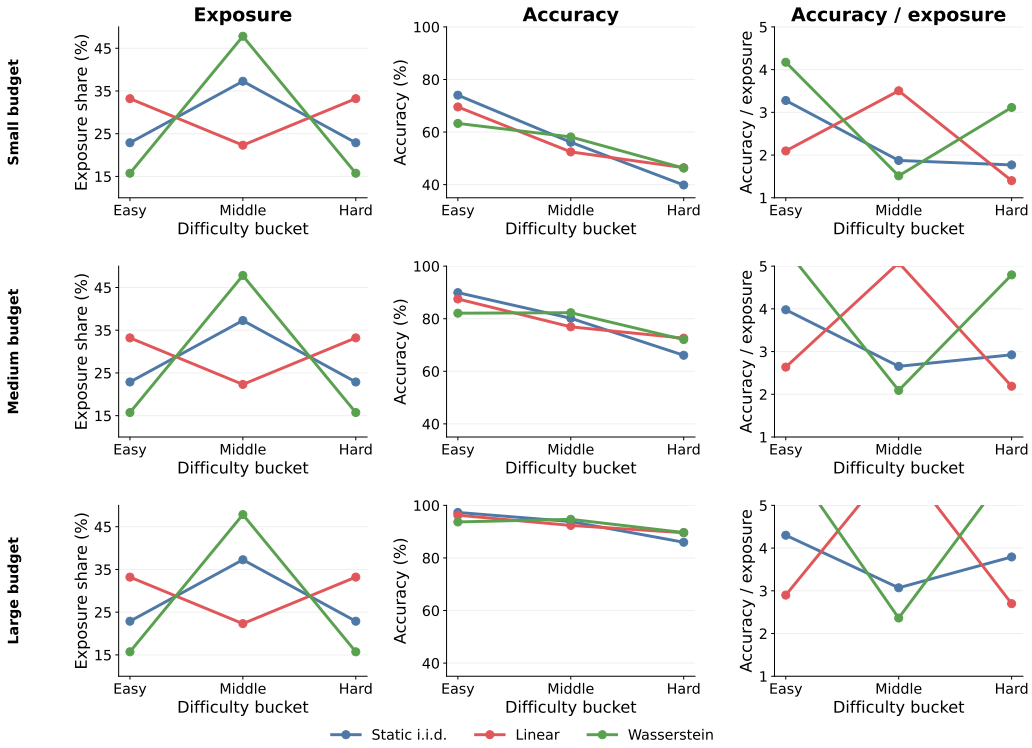


Figure 21: Exposure, accuracy, and exposure-adjusted accuracy across budgets. Rows denote the small, medium, and large budgets. Columns denote cumulative exposure share, final bucket accuracy, and exposure-adjusted accuracy on the easiest, middle, and hardest probe buckets.

C.3 Task-Wise Final Level Profiles

Setup. To complement the task-wise endpoint barplots above, we also visualize where final performance lands across the available difficulty levels. Each figure again corresponds to one task, with rows denoting difficulty axes and columns denoting the small, medium, and large budgets. Within each panel, colored curves trace the mean final accuracy at each level for the eight curricula. Tasks with four levels stop at Level 4, while five-level tasks extend through Level 5.

Findings. These profile grids make the location of gains explicit: some curricula preserve more of the easy-end accuracy, while others shift the profile upward at the harder end. They therefore complement the budget-wise summaries and task-wise endpoint barplots by showing not only which curriculum wins overall, but also where along the task's level structure the final gains and tradeoffs occur.

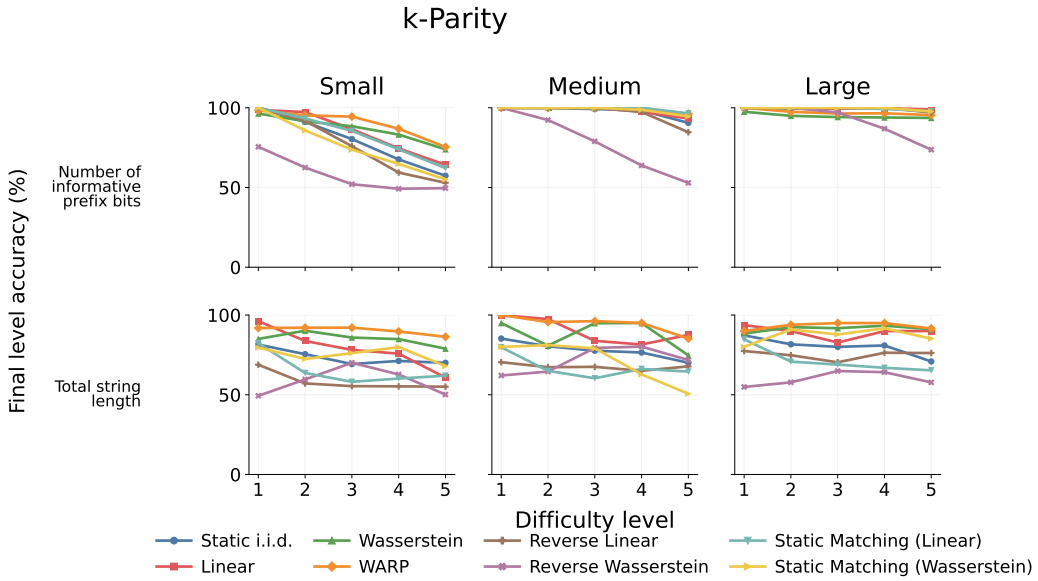


Figure 22: Task-wise final level profiles for K-Parity. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

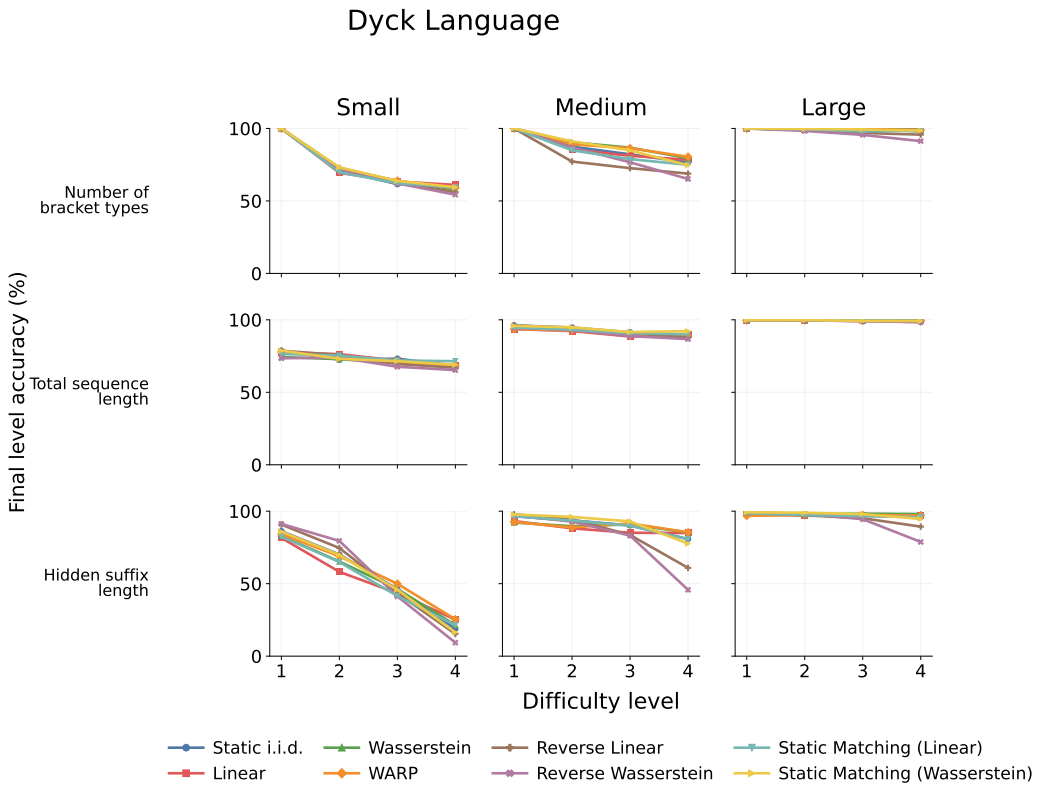


Figure 23: Task-wise final level profiles for Dyck Language. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

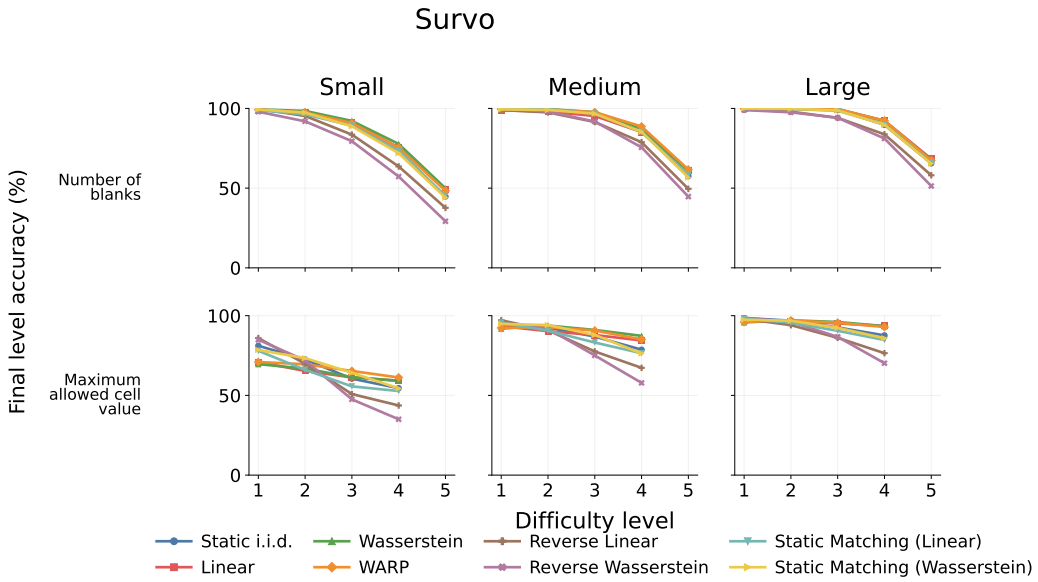


Figure 24: Task-wise final level profiles for Survo. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

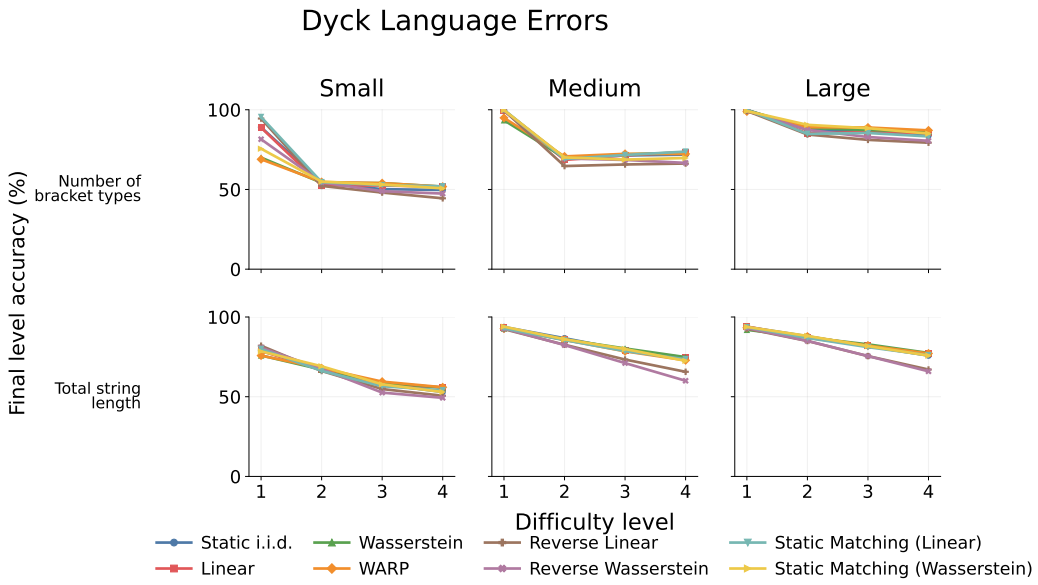


Figure 25: Task-wise final level profiles for Dyck Language Errors. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

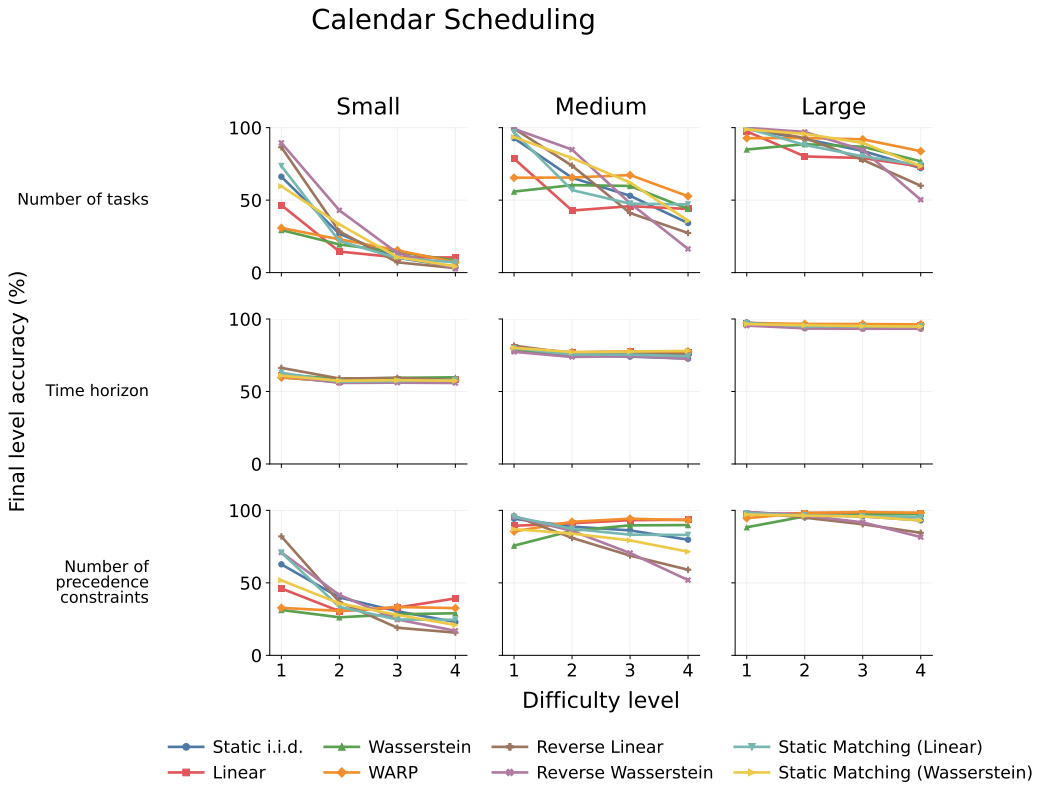


Figure 26: Task-wise final level profiles for Calendar Scheduling. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

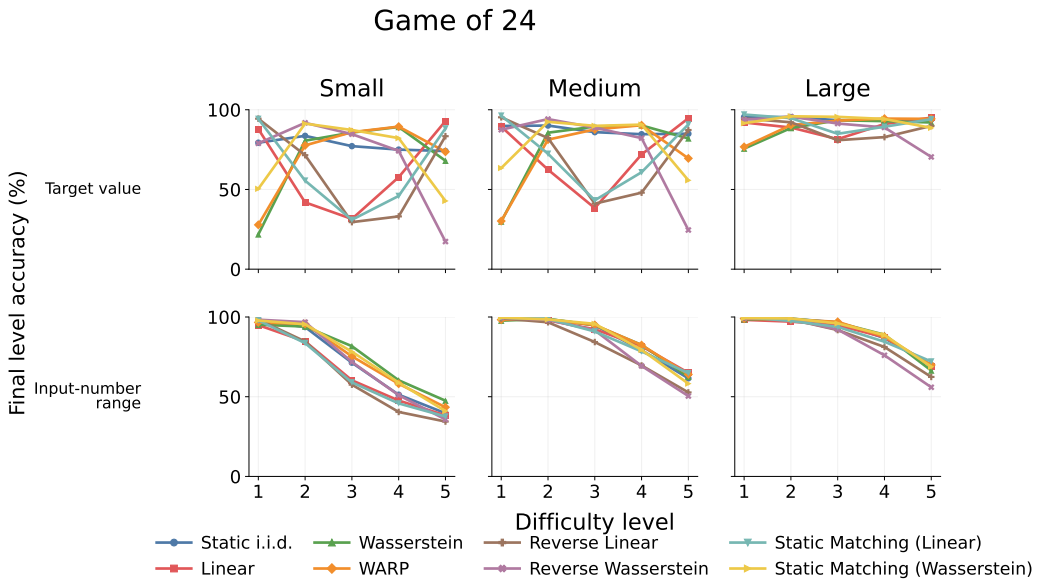


Figure 27: Task-wise final level profiles for Game of 24. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

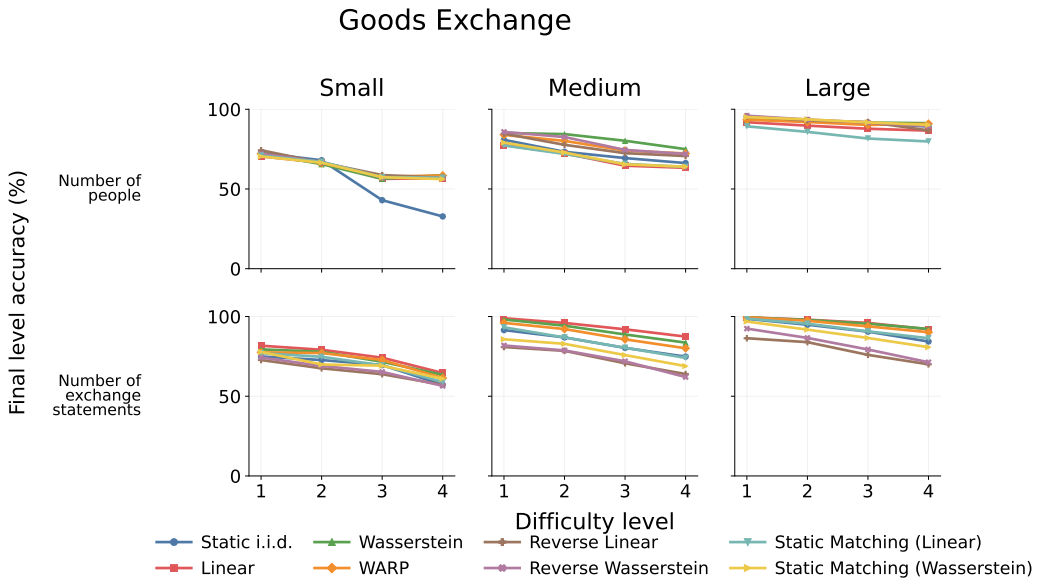


Figure 28: Task-wise final level profiles for Goods Exchange. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

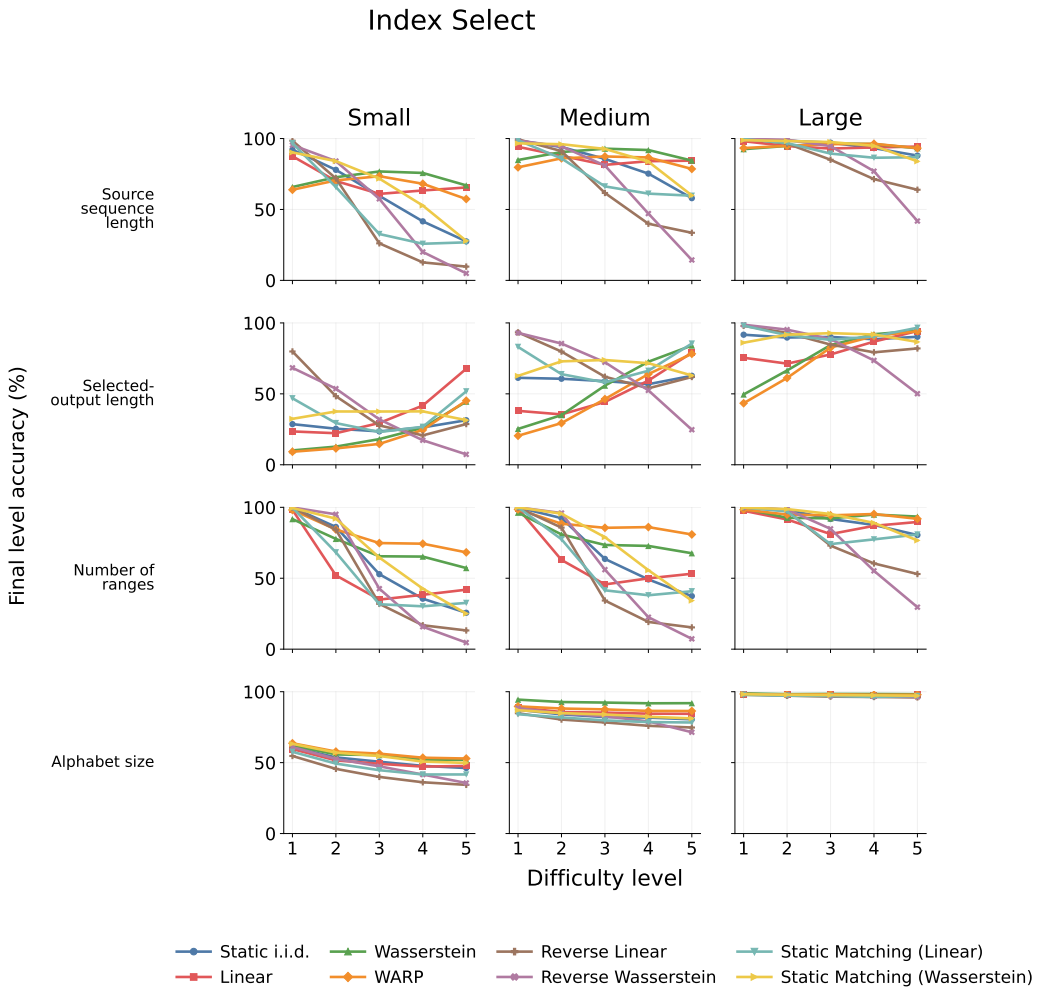


Figure 29: Task-wise final level profiles for Index Select. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

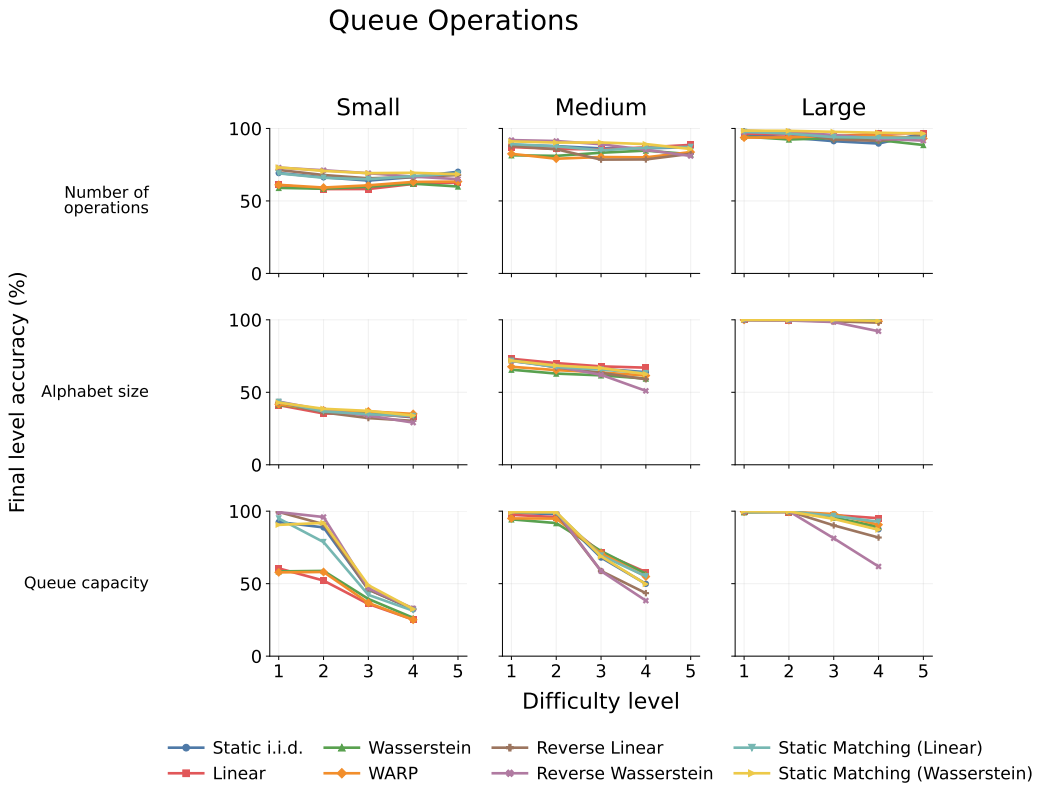


Figure 30: Task-wise final level profiles for Queue Operations. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

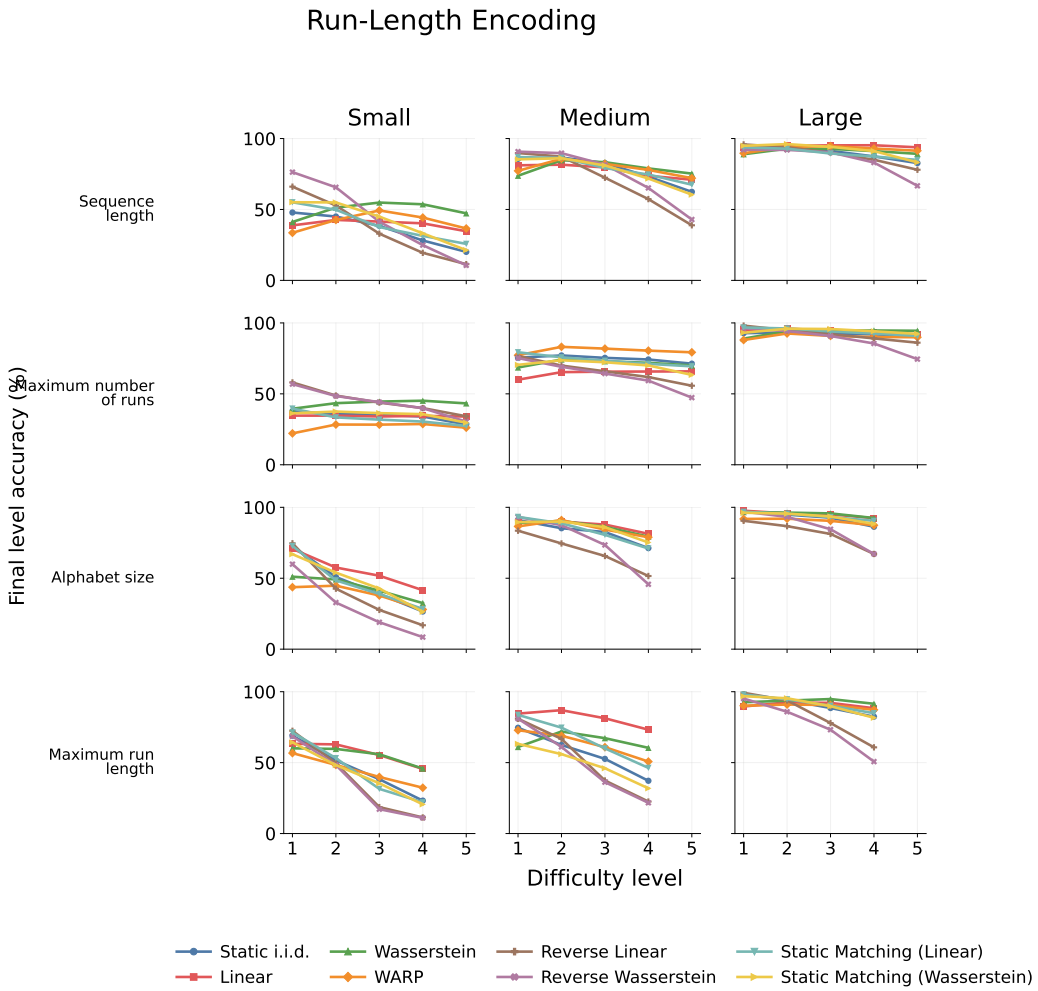


Figure 31: Task-wise final level profiles for Run-Length Encoding. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

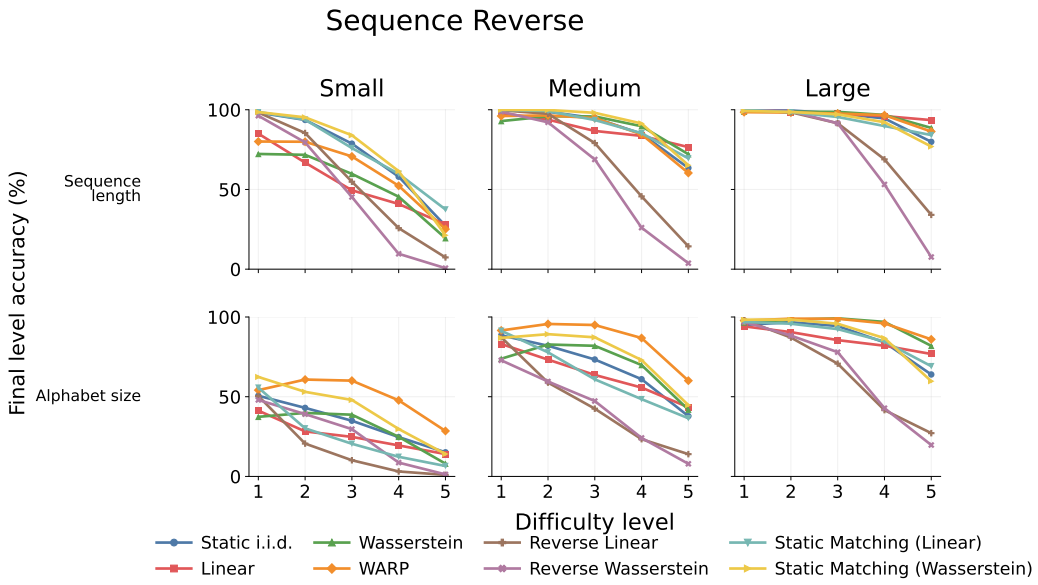


Figure 32: Task-wise final level profiles for Sequence Reverse. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

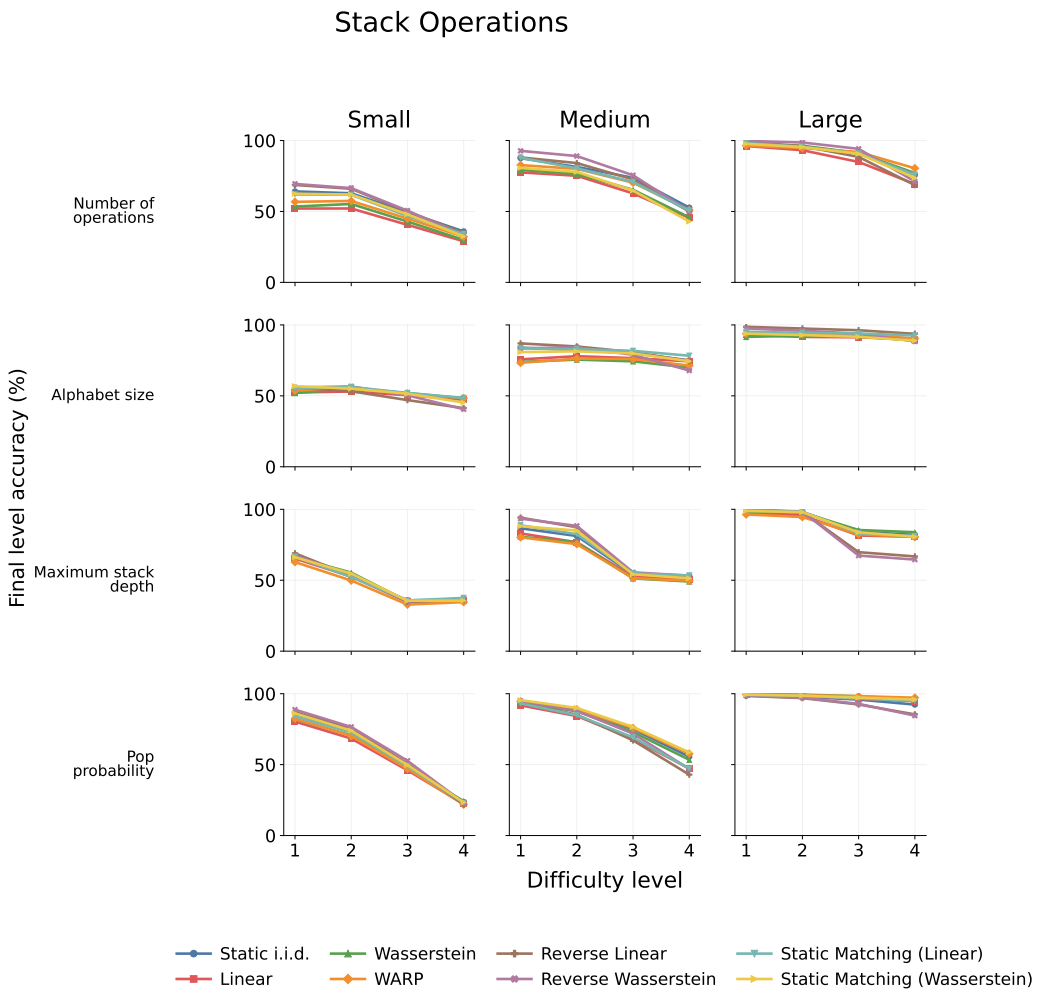


Figure 33: Task-wise final level profiles for Stack Operations. Rows denote difficulty axes, columns denote budgets, and curves show mean final accuracy at each available difficulty level over 10 random-seed repetitions.

C.4 Exploratory Analysis of When Easy-to-Hard Progression Helps

Many curriculum methods depend strongly on the chosen difficulty signal or decomposition (Platanios et al., 2019; Hacohen & Weinshall, 2019; Swayamdipta et al., 2020; Jia et al., 2026). We group the 99 task-by-axis-and-budget conditions by task family, difficulty-axis category, and their intersection, and count where *Wasserstein* or WARP attains the best mean among the four forward curricula.

I. Tasks. We first ask the same question at the task level by grouping the 12 benchmarks into four coarse task families listed below. Unlike the difficulty-axis taxonomy below, these task families do not overlap.

- **Sequence Transformation:** INDEX SELECT, RUN-LENGTH ENCODING, SEQUENCE REVERSE
- **Formal Structure:** DYCK LANGUAGE, DYCK LANGUAGE ERRORS, K-PARITY
- **Data-Structure Execution:** QUEUE OPERATIONS, STACK OPERATIONS
- **Constraint and Search:** CALENDAR SCHEDULING, GAME OF 24, GOODS EXCHANGE, SURVO

Task family	Overall wins	Hardest-level wins
Sequence Transformation	18/30 (10 W, 8 A)	21/30 (14 W, 7 A)
Formal Structure	15/21 (4 W, 11 A)	13/21 (4 W, 9 A)
Data-Structure Execution	7/21 (2 W, 5 A)	8/21 (2 W, 6 A)
Constraint and Search	15/27 (7 W, 8 A)	15/27 (8 W, 7 A)

Table 4: Task-family summary of when natural easy-to-hard progression helps. We group the 12 benchmarks into four coarse task families and count the number of task-by-axis-and-budget conditions for which *Wasserstein* or WARP attains the best mean among the four forward curricula (*static i.i.d.*, linear, Wasserstein, WARP). The left numeric pair reports overall wins; the right reports wins on the hardest available level. 'W' denotes Wasserstein and 'A' denotes WARP.

Task findings. The task taxonomy is somewhat cleaner than the axis taxonomy, but it still does not yield a single simple rule. Natural easy-to-hard progression is strongest on *Sequence Transformation* tasks, where it wins in a majority of conditions overall and even more often on the hardest level. *Formal Structure* and the broader *Constraint and Search* family also show substantial support, especially at the hard end. By contrast, *Data-Structure Execution* remains more mixed.

II. Difficulty Axes. We next simplify the raw axis list by grouping the axes into four broad categories: *Context Length* (how much input must be processed), *Answer Length* (how much output must be produced), *Entity Complexity* (the number or variety of entities/attributes), and *Procedural Complexity* (the depth or expressivity of the reasoning chain). We again examine winners both in final overall accuracy and on the hardest available level.

- **Context Length:** representative pairs include CALENDAR SCHEDULING / *Time horizon*, DYCK LANGUAGE / *Total sequence length*, INDEX SELECT / *Source sequence length*, and SURVO / *Number of blanks*
- **Answer Length:** representative pairs include DYCK LANGUAGE / *Hidden suffix length*, INDEX SELECT / *Selected-output length*, RUN-LENGTH ENCODING / *Maximum run length*, and SURVO / *Number of blanks*
- **Entity Complexity:** representative pairs include DYCK LANGUAGE / *Number of bracket types*, GAME OF 24 / *Input-number range*, INDEX SELECT / *Alphabet size*, and SURVO / *Maximum allowed cell value*
- **Procedural Complexity:** representative pairs include CALENDAR SCHEDULING / *Number of precedence constraints*, GOODS EXCHANGE / *Number of exchange statements*, INDEX SELECT / *Number of ranges*, and STACK OPERATIONS / *Maximum stack depth*

Axis category	Overall wins	Hardest-level wins
Context Length	25/39 (12 W, 13 A)	23/39 (13 W, 10 A)
Answer Length	10/18 (6 W, 4 A)	15/18 (11 W, 4 A)
Entity Complexity	26/39 (10 W, 16 A)	23/39 (8 W, 15 A)
Procedural Complexity	11/30 (3 W, 8 A)	16/30 (8 W, 8 A)

Table 5: Difficulty-axis-category summary of when natural easy-to-hard progression helps. We group difficulty axes into four broad categories and count the number of task-by-axis-and-budget conditions for which *Wasserstein* or WARP attains the best mean among the four forward curricula (*static i.i.d.*, linear, Wasserstein, WARP). The left numeric pair reports overall wins; the right reports wins on the hardest available level. 'W' denotes Wasserstein and 'A' denotes WARP. Because some axes belong to more than one category, the denominators overlap across rows.

Axis findings. The axis taxonomy helps simplify the picture, but it still does not collapse the results to a single winning axis type. *Context Length* and *Entity Complexity* contain many overall wins, suggesting that natural easy-to-hard progression often helps when an axis enlarges the amount of input/state to process or the variety of entities/attributes.

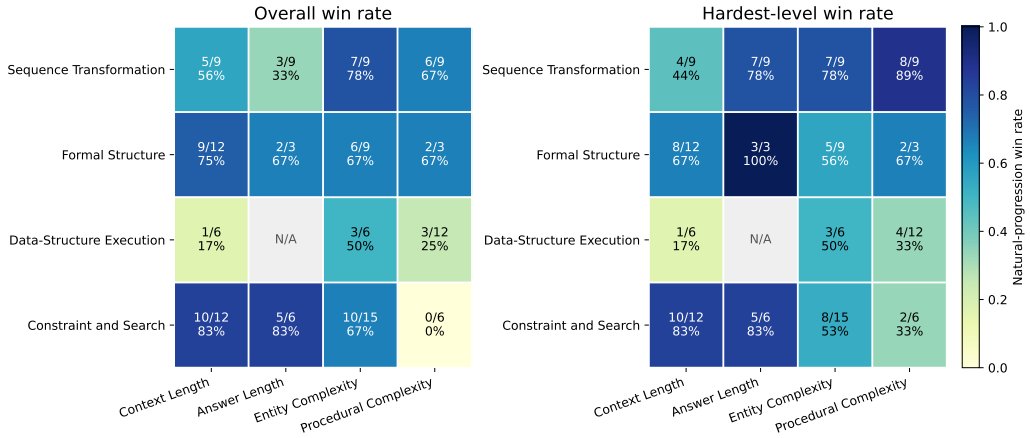


Figure 34: Joint task-family × difficulty-axis-category view of when natural easy-to-hard progression helps. Rows denote task families and columns denote difficulty-axis categories. The left panel shows overall win rates and the right panel shows hardest-level win rates for *Wasserstein* or WARP against the other forward curricula. Each cell reports both the raw count and the percentage. Gray cells indicate that no task in that family has an axis of the corresponding category.

Answer Length is the strongest category on the hardest level (15/18), driven by axes such as *Hidden suffix length*, *Maximum run length*, and *Selected-output length*. *Procedural Complexity* is more mixed: it is the weakest category overall, but its hardest-level rate improves substantially, suggesting that easy-to-hard progression can still help on the difficult end without consistently improving aggregate performance.

III. Joint View. Figure 34 combines the two taxonomies by showing, for each task family and difficulty-axis category pair, the fraction of conditions in which *Wasserstein* or WARP attains the best mean among the four forward curricula. The left panel reports overall wins and the right reports hardest-level wins; each cell is annotated with both the raw count and the corresponding win rate.

Joint findings. The interaction view clarifies where the previous summaries line up and where they do not. *Sequence Transformation* is strongest when paired with *Entity Complexity* or *Procedural Complexity* axes, especially on the hardest level. *Constraint and Search* is strongest when paired with *Context Length* or *Answer Length* axes, but shows little support on *Procedural Complexity*. *Formal Structure* remains broadly positive across categories, though with fewer total conditions, while *Data-Structure Execution* stays weak across most cells. These three views suggest that easy-to-hard progression is easier to characterize at the level of coarse task family than at the level of semantic axis type, but even the joint taxonomy remains only partially predictive.

The taxonomy views above describe where natural easy-to-hard progression can help. We next ask what mechanisms may explain those gains by examining isolated level-to-level transfer and controlled bridge effects.

C.5 Level Transfer Experiments

We study isolated transfer between difficulty levels to ask a simple mechanism question: when source-level training helps a target level, does it mainly give the target stage a *head start*, or does it make target-stage learning itself *faster*?

Setup. For each task, difficulty axis, seed, and ordered pair of distinct levels, we train on a source level and then continue only on a destination level. Let B denote the task's per-level small budget used in the main synthetic experiments. We write $s \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$

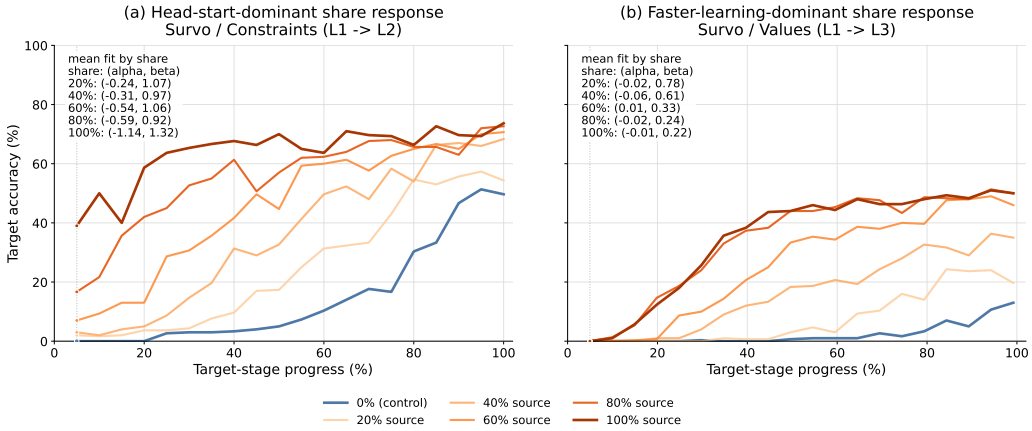


Figure 35: Two easy-to-hard examples under the first-stage fit $t_{\mathrm{tr},i}(a;s) \approx \alpha_{i,s} + \beta_{i,s}\, t_{\mathrm{ctrl},i}(a)$. We define $t(a)$ as the first time a target-accuracy curve reaches a. Blue: matched 0% source-share control. Orange shades: transferred runs with source shares $s \in \{20, 40, 60, 80, 100\}$%. Curves start at the first target-stage evaluation, at 5% progress. Panel (a) is head-start dominant. Panel (b) is faster-learning dominant. All axes are in percent.

for the source-stage share, so the source stage uses sB steps while the destination stage always uses B steps. We compare each transferred run against the matched $s = 0$ no-transfer control. We start with lower-to-higher transfer and then return to higher-to-lower transfer when analyzing reverse ordering.

Fit by source share. For each easy-to-hard ordered pair i and each fixed $s > 0$, let $A_{\mathrm{tr},i}(t;s)$ denote target accuracy during the target stage, where $t \in [0,1]$ is normalized target-stage progress. Let $A_{\mathrm{ctrl},i}(t) = A_{\mathrm{tr},i}(t;0)$ denote the matched no-transfer control. Over the accuracy range reached by both runs, we define $t_{\mathrm{tr},i}(a;s)$ and $t_{\mathrm{ctrl},i}(a)$ as the earliest target-stage progress at which each run first reaches accuracy a, using linear interpolation between evaluations. We first fit

$$t_{\mathrm{tr},i}(a;s) \approx \alpha_{i,s} + \beta_{i,s}\, t_{\mathrm{ctrl},i}(a).$$

This gives one fitted pair $(\alpha_{i,s}, \beta_{i,s})$ for each ordered pair and each source share. We then model how these coefficients vary with source share:

$$\alpha_{i,s} = u_{\alpha,i} + f_\alpha(s), \qquad \log \beta_{i,s} = u_{\beta,i} + f_\beta(s),$$

where $u_{\alpha,i}$ and $u_{\beta,i}$ are pair-specific effects and f_α, f_β are quadratic functions of s. We anchor the fit at the no-transfer point $(s = 0, \alpha = 0, \beta = 1)$. Negative $\alpha_{i,s}$ indicates a head start. Values $\beta_{i,s} < 1$ indicate faster target-stage learning. Figure 35 shows two representative first-stage examples. Figure 36 shows how the fitted coefficients vary with source share, together with the empirical distributions.

Results. Figure 35 shows two recurring patterns: larger source share either shifts the target-stage curve earlier or steepens the remaining climb. Figure 36 shows the same pattern in aggregate. Across all easy-to-hard pairs, the fitted coefficients move from $(-0.08, 0.83)$ at 20% source share to $(-0.43, 0.45)$ at 100%, so both the head-start effect and the faster-learning effect grow with source share. On hardest-target pairs, the pattern leans further toward faster learning: the fitted coefficients move from $(-0.07, 0.85)$ at 20% to $(-0.12, 0.36)$ at 100%. The high end also flattens: on this subset, mean β changes from 0.63 to 0.55 between 80% and 100%, while mean α changes from -0.15 to -0.11.

Finding 5. *Easy-to-hard transfer helps through both head start and faster target-stage learning. Across source shares, faster target-stage learning is especially prominent on the hardest targets.*

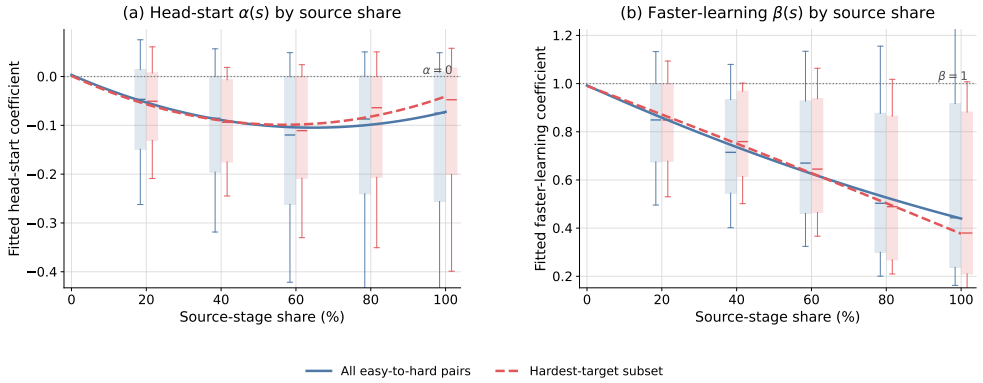


Figure 36: Transfer coefficients by source share. We estimate $(\alpha_{i,s}, \beta_{i,s})$ at each observed source share from earliest hitting times on target-accuracy curves, then fit quadratic trends to the median coefficients across shares (anchored at the no-transfer point). Curves show the median-by-share trend and boxplots show the empirical distribution. Solid blue uses all easy-to-hard ordered pairs. Dashed red uses the hardest-target subset.

Directional asymmetry. Using isolated transfer at 100% source share, we compare each lower/higher level pair in both directions. Figure 37(a) shows mean final target-accuracy gain over the matched 0% control; hard-to-easy transfer is often larger, especially toward easy targets. Panels (b) and (c) show why reverse still loses: it spends early budget on hard levels, leaving less remaining margin when it reaches easier targets, so the Wasserstein-over-reverse gap is small (often negative) on the easy bucket and large on the hard bucket. Reverse's extra easy-bucket gains therefore do not offset Wasserstein's larger hard-bucket gains.

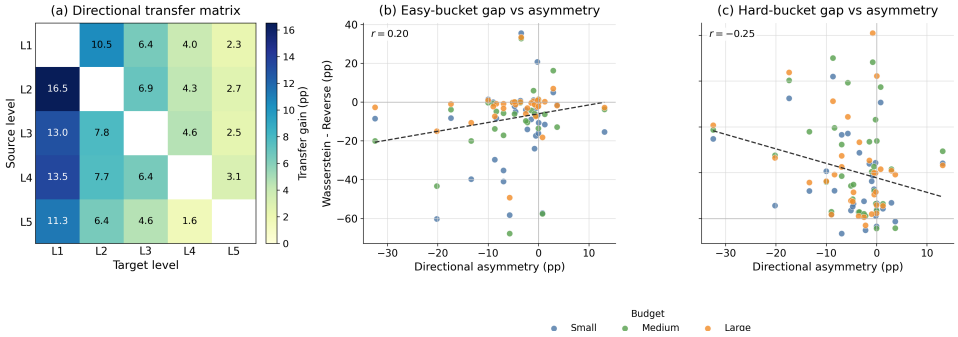


Figure 37: Directional asymmetry in isolated transfer at 100% source share. Panel (a) shows mean final target-accuracy gain over matched 0% controls for each ordered pair; the upper triangle is easy-to-hard and the lower triangle is hard-to-easy. Panels (b) and (c) show Wasserstein minus reverse gaps on the easy and hard buckets as a function of final-gain asymmetry.

Finding 6. *Hard-to-easy transfer is often larger than easy-to-hard transfer, but reverse Wasserstein still loses: it trains on hard levels first and reaches easier targets later with less remaining margin, so easy-bucket gains do not offset Wasserstein's larger hard-bucket gains.*

C.6 Bridge Effect Experiments

Setup. We test whether an intermediate level helps a hard target by running three-stage experiments on contiguous triples $(\ell, \ell+1, \ell+2)$. Stage 1 trains on the source level, stage

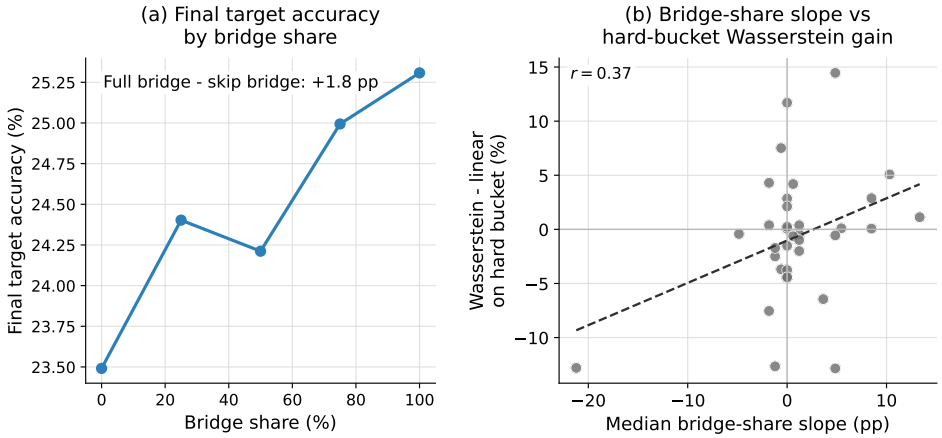


Figure 38: Bridge-share response experiments. Panel (a) shows mean final target accuracy versus bridge share, averaged over bridge triples and seeds. Panel (b) plots, for each task-by-axis setting, the median bridge-share slope over hardest-target triples against the medium-budget hard-bucket Wasserstein–linear gap.

2 uses a fixed refine budget that is split between staying on the source and moving to the bridge level, and stage 3 trains on the target level. We write $b \in \{0, 0.25, 0.5, 0.75, 1.0\}$ for the fraction of the stage-2 budget spent on the bridge level, where $b = 0$ is the skip-bridge control and $b = 1$ is the full-bridge condition. For triple i, let $A_i(b)$ denote final target accuracy. To use the full bridge-share sweep, we fit

$$A_i(b) \approx u_i + \lambda_i b,$$

where u_i is a triple-specific intercept and λ_i is the bridge-share slope. Larger λ_i means that moving more of stage 2 onto the bridge level helps more.

Results. Figure 38(a) shows an overall upward response: mean final target accuracy rises from 23.49% at 0% bridge share to 25.31% at 100%, though the sweep is not perfectly monotone. Figure 38(b) then plots, for each task-by-axis setting, the median bridge-share slope over hardest-target triples against the medium-budget hard-bucket Wasserstein–linear gap from Section 4.1. The association is positive ($r = 0.37$). Settings whose hard targets improve more as bridge share increases also tend to be the settings where Wasserstein gains more over linear on the hard bucket. This is consistent with the same local-transfer picture suggested by Figure 37(a): an intermediate level can split one hard jump into two easier ones.

Finding 7. *Larger bridge effects tend to align with larger Wasserstein-over-linear gains.*

D Real-World SFT Analyses

This appendix studies the pretrained SFT setting. We first evaluate real-world SFT across three base models, then use a synthetic-pretraining analogue to test whether pretraining itself weakens curriculum effects.

D.1 Real-World SFT Benchmarks

Setup. We run LoRA (Hu et al., 2022) SFT on ARC (Clark et al., 2018), MMLU-STEM (the STEM subset of MMLU; Hendrycks et al. 2021), and StrategyQA (Geva et al., 2021), following the benchmark construction and difficulty labels from Hase et al. (2024). We

compare `SmolLM3-3B`[1], `OLMo2-1B`[2], and `Qwen3-1.7B`[3] under the same curriculum sweep, with three seeds, learning rate 5×10^{-6}, and batch size 128.

Benchmarks and difficulty axes. We study 12 benchmark-by-axis contexts: six for ARC, three for MMLU-STEM, and three for StrategyQA. ARC uses *human grade* (6 levels), *human difficulty* (3), *human Bloom level* (5), *human depth of knowledge* (3), *question length* (5 quantile bins), and *answer length* (5 quantile bins). MMLU-STEM uses *human hardness* (2 levels), *question length* (5 quantile bins), and *answer length* (5 quantile bins) over the five STEM subject groups from Hase et al. (2024). StrategyQA uses *decomposition length* (5 levels), *question length* (5 quantile bins), and *reasoning length* (5 quantile bins). Training targets contain a short reasoning chain plus the final answer, while evaluation uses only the final answer.

Budgets. We calibrate small, medium, and large budgets separately for each dataset: ARC uses 32, 64, and 128 update steps; MMLU-STEM uses 16, 32, and 64; and StrategyQA uses 64, 96, and 128. Unless otherwise stated, we average over the same three random seeds.

Static i.i.d. Remains the Strongest Summary Baseline. Tables 6, 7, and 8 summarize the three basic curricula over these 12 contexts for each model. Across all three models, *static i.i.d.* has the highest mean overall accuracy and the highest mean hardest-level accuracy in every model-budget block. Linear and Wasserstein still win isolated contexts, but the gaps are small and much less systematic than in the synthetic suite.

[1] https://huggingface.co/HuggingFaceTB/SmolLM3-3B-Base
[2] https://huggingface.co/allenai/OLMo-2-0425-1B
[3] https://huggingface.co/Qwen/Qwen3-1.7B-Base

Curriculum	Mean overall (%)	Mean hardest (%)	Overall wins	Hardest wins
Small budget				
Static i.i.d.	**57.5**	**53.7**	**8**	**8**
Linear	57.0	52.5	3	3
Wasserstein	57.1	52.7	1	1
Medium budget				
Static i.i.d.	**63.4**	**58.3**	**6**	**6**
Linear	62.7	57.3	3	1
Wasserstein	62.7	58.1	3	5
Large budget				
Static i.i.d.	**67.6**	**63.4**	**9**	**6**
Linear	66.0	62.1	1	4
Wasserstein	66.6	62.9	2	2

Table 6: SFT summary for the three basic curricula on ARC, MMLU-STEM, and StrategyQA using `SmolLM3-3B`. Each budget block contains 12 benchmark-by-axis contexts: six ARC axes, three MMLU-STEM axes, and three StrategyQA axes. "Hardest" denotes the last available level.

Curriculum	Mean overall (%)	Mean hardest (%)	Overall wins	Hardest wins
Small budget				
Static i.i.d.	**37.6**	**35.5**	4	**6**
Linear	37.4	35.4	3	3
Wasserstein	37.5	35.5	**5**	3
Medium budget				
Static i.i.d.	**39.0**	**36.8**	**6**	**5**
Linear	38.2	36.1	1	3
Wasserstein	38.5	36.4	5	4
Large budget				
Static i.i.d.	**45.6**	**42.8**	**8**	**6**
Linear	44.4	41.6	3	5
Wasserstein	44.4	41.3	1	1

Table 7: SFT summary for the three basic curricula on ARC, MMLU-STEM, and StrategyQA using `OLMo2-1B`. Each budget block contains 12 benchmark-by-axis contexts: six ARC axes, three MMLU-STEM axes, and three StrategyQA axes. "Hardest" denotes the last available level.

Curriculum	Mean overall (%)	Mean hardest (%)	Overall wins	Hardest wins
Small budget				
Static i.i.d.	**62.6**	**60.4**	**6**	**5**
Linear	62.1	60.2	3	3
Wasserstein	61.8	60.4	3	4
Medium budget				
Static i.i.d.	**68.6**	**66.4**	**5**	**5**
Linear	68.1	66.0	4	**5**
Wasserstein	68.1	65.7	3	2
Large budget				
Static i.i.d.	**71.4**	**67.8**	**7**	**6**
Linear	70.5	67.4	4	4
Wasserstein	70.6	67.4	1	2

Table 8: SFT summary for the three basic curricula on ARC, MMLU-STEM, and StrategyQA using `Qwen3-1.7B`. Each budget block contains 12 benchmark-by-axis contexts: six ARC axes, three MMLU-STEM axes, and three StrategyQA axes. "Hardest" denotes the last available level.

D.2 Final Level Distributions, Bucket Profiles, and Remaining Efficiency Differences

Appendix D.1 showed that *static i.i.d.* is the strongest summary baseline in pretrained SFT. We next ask whether the forward curricula still differ in exposure allocation and exposure-adjusted accuracy.

Setup. We study the medium budget and the four forward curricula: *static i.i.d.*, linear, Wasserstein, and WARP. To keep the level view direct, we restrict to the 10 contexts whose native difficulty axes already have 3 or 5 levels, excluding only ARC *human grade* (6 levels) and MMLU-STEM *human hardness* (2 levels). Thus 3-level axes map directly to *easy/middle/hard*, and 5-level axes to *easy/lower-mid/middle/upper-mid/hard*. We report both five-position exposure/accuracy summaries and bucketed easy/middle/hard summaries with exposure-adjusted accuracy.

Figure 39 shows the five-position view. Exposure shifts much more than accuracy: linear pushes exposure toward the ends, while Wasserstein and WARP push it toward the middle, but the final accuracy curves remain close. WARP also stays close to fixed Wasserstein.

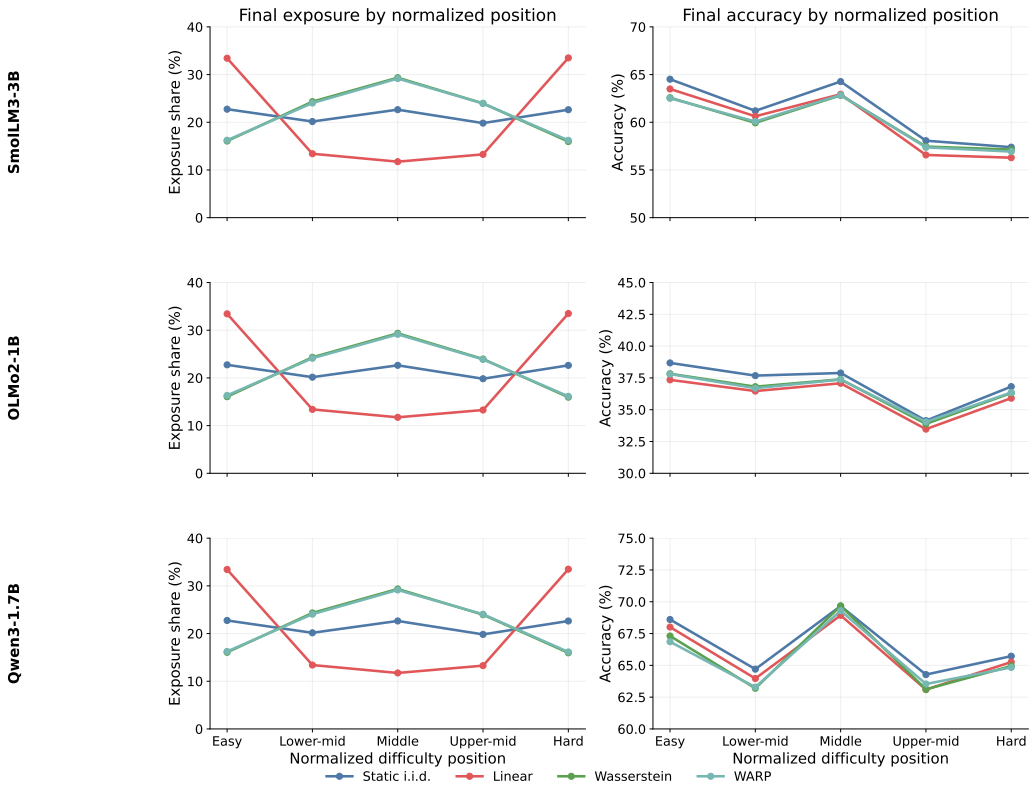


Figure 39: Medium-budget SFT exposure and final accuracy on the 10 retained 3- or 5-level contexts. Rows show the three models; curves show the four forward curricula.

Figure 40 shows the bucketed view. Raw bucket accuracies remain compressed despite large exposure differences, so the exposure-adjusted view is more informative. Linear is strongest in the middle bucket, while Wasserstein and WARP are strongest on the hard bucket, but the gaps are modest and model-dependent.

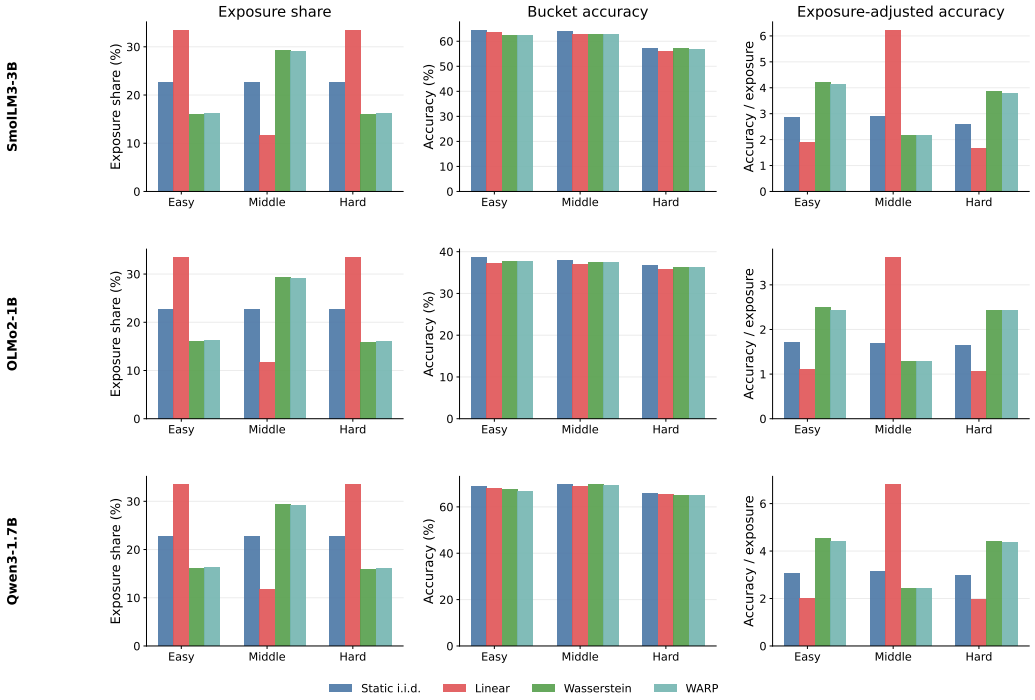


Figure 40: Bucket-wise exposure, final accuracy, and exposure-adjusted accuracy for medium-budget SFT on the same 10 retained contexts. Rows show the three models; columns show exposure, accuracy, and exposure-adjusted accuracy under the four forward curricula.

D.3 Easy-to-Hard Ordering Effect Is Weak in Pretrained SFT

Appendix D.2 showed that pretrained SFT largely flattens the forward-curriculum profiles. We next ask whether any ordering signal survives once exposure geometry is held fixed.

Setup. Figure 41 compares three Wasserstein-family controls across the small, medium, and large budgets: forward Wasserstein, reverse Wasserstein, and the exposure-matched static baseline. We keep all 12 benchmark-by-axis contexts and report easy, middle, and hard bucket means for each model.

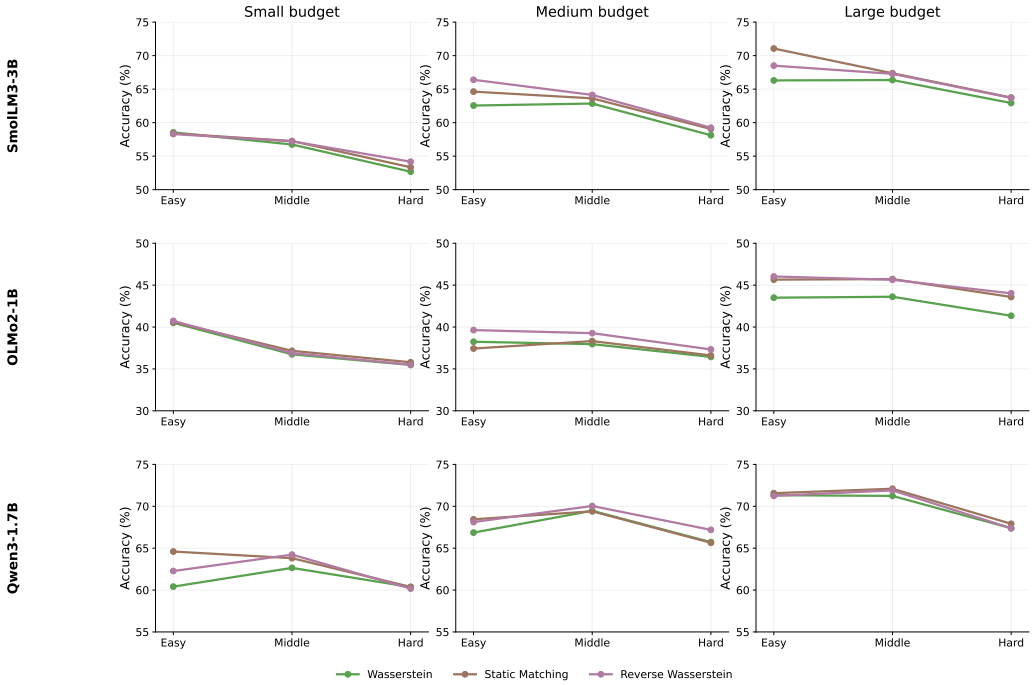


Figure 41: Ordering comparison within the Wasserstein family for SFT on ARC, MMLU-STEM, and StrategyQA. Rows show the three models; columns show the small, medium, and large budgets. "Static Matching" denotes the exposure-matched static baseline.

Forward ordering is no longer distinctly stronger. Across all nine model-by-budget blocks, forward Wasserstein never has the highest hard-bucket mean; the exposure-matched static baseline or the reverse-order control is always as good or better. Under pretrained SFT, changing the ordering within a fixed geometry no longer yields a robust easy-to-hard advantage.

Finding 8. *Under pretrained SFT, most curriculum effects seen in the synthetic study weaken: static i.i.d. becomes the strongest summary baseline, bucket accuracies flatten, and easy-to-hard ordering no longer separates reliably. The clearest remaining difference is in exposure-adjusted data efficiency.*

D.4 Exploratory Analysis of When Easy-to-Hard Progression Helps

We next ask whether the remaining easy-to-hard wins cluster by benchmark or difficulty-axis type.

Setup. We analyze conditions in which *Wasserstein* or WARP attains the best mean among the four forward curricula, counting ties for all tied curricula under each metric. We then summarize these wins by benchmark and by difficulty-axis type.

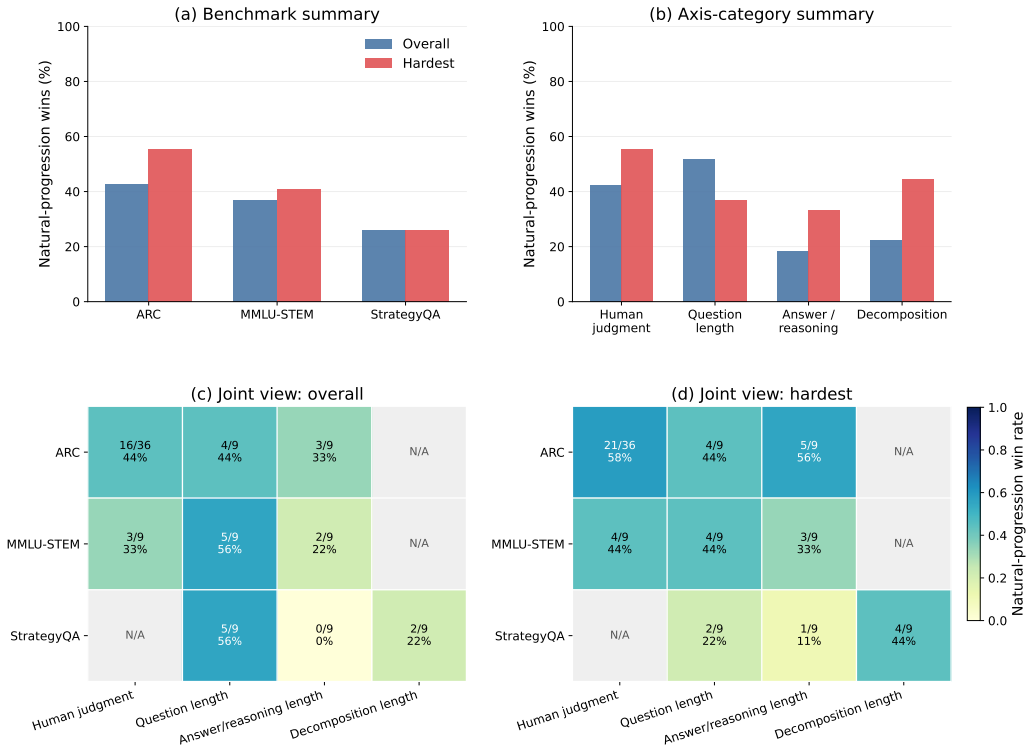


Figure 42: Where natural easy-to-hard progression helps in SFT. Panel (a) summarizes wins by benchmark, panel (b) by axis category, and panels (c)–(d) show the joint benchmark × axis-category view for overall and hardest-level wins.

The remaining easy-to-hard wins are concentrated on ARC and human-annotated difficulty axes. Figure 42 shows only a weak concentration of wins. ARC has the most wins, especially on hardest-level accuracy, and human-annotated difficulty axes have the highest hardest-level win rate among the axis categories. However, neither pattern is consistent across both overall and hardest-level accuracy. Unlike Appendix C.4, the SFT setting does not yield a stable benchmark- or axis-level pattern.

D.5 Controlled Synthetic Pretraining Analogue

To isolate the effect of pretraining, we vary the pretraining budget on the synthetic suite while holding the downstream tasks and evaluation protocol fixed.

Setup. In the pretraining stage, we train a shared 6-layer decoder on nine synthetic tasks from Appendix B.1. We use a uniform multitask mixture that splits each task's small-budget quota evenly across its reported difficulty axes and levels, then globally shuffles the resulting examples. We study three pretraining regimes, *small*, *medium*, and *large*, corresponding to 20%, 50%, and 100% of the sum of the task-wise small budgets. Downstream, we keep the earlier synthetic setup, restrict the sweep to the small budget, and remove pretraining-seen examples from the downstream train split.

Aggregate gaps narrow as pretraining grows. Table 9 shows the aggregate picture. As pretraining increases, the three basic curricula become harder to distinguish. The static baseline strengthens substantially: its mean hardest-level accuracy rises from 84.7% under small pretraining to 91.5% under medium pretraining and 93.7% under large pretraining. Even so, unlike the SFT setting, static does not become the uniformly strongest hardest-level baseline. Linear and Wasserstein continue to match or exceed it on many contexts.

Curriculum	Mean overall (%)	Mean hardest (%)	Overall wins	Hardest wins
Small pretraining				
Static i.i.d.	**91.3**	84.7	10	6
Linear	91.2	**88.1**	**13**	**17**
Wasserstein	91.2	87.1	9	11
Medium pretraining				
Static i.i.d.	95.0	91.5	13	13
Linear	95.0	**92.9**	**17**	**18**
Wasserstein	95.0	92.3	10	11
Large pretraining				
Static i.i.d.	96.0	93.7	**19**	**18**
Linear	96.4	**94.6**	16	**18**
Wasserstein	**96.5**	**94.6**	14	14

Table 9: Synthetic-pretraining summary for the three basic curricula under a fixed small downstream budget. Each pretraining block contains the same 26 synthetic task-by-axis contexts. Here, "large pretraining" denotes the full multitask pretraining budget. The win columns count the number of contexts in which a curriculum attains the best mean under the corresponding metric; ties are counted for all tied curricula. "Hardest" denotes the last available level.

Level-wise profiles flatten as pretraining grows. Figure 43 shows the same narrowing in level-wise profiles. Under *static i.i.d.*, the mean easy–hard gap falls from 10.9 points under small pretraining to 5.9 under medium pretraining and 4.8 under large pretraining; under Wasserstein, it shrinks from 5.5 to 4.5 to 3.3. This mirrors the SFT trend, though part of the narrowing here comes from easy-level saturation near 100%. Even so, the exposure-adjusted pattern survives: linear remains strongest on the middle bucket, while Wasserstein remains strongest on the hard bucket.

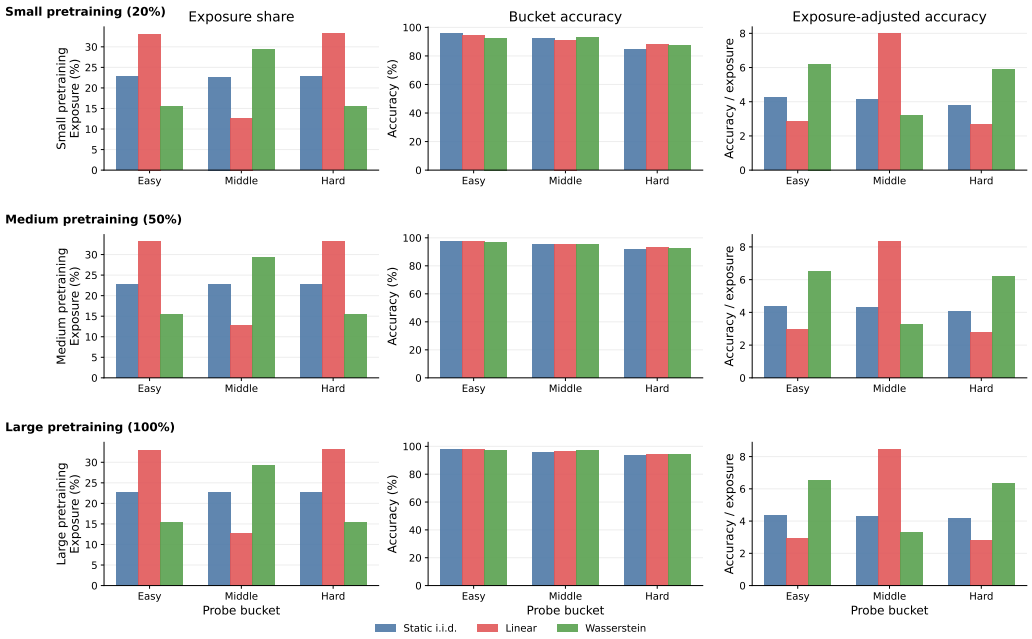


Figure 43: Synthetic-pretraining analogue under a fixed small downstream budget. Rows vary pretraining amount; columns show level-wise exposure, final accuracy, and exposure-adjusted accuracy for *static i.i.d.*, linear, and Wasserstein.

Ordering weakens, but does not disappear. Figure 44 shows that the easy-to-hard advantage within the Wasserstein family shrinks steadily as pretraining increases. The Wasserstein–reverse gap falls from 13.5 points under small pretraining to 2.3 under large pretraining, while the Wasserstein–exposure-matched-static gap falls from 3.9 to 0.2. The corresponding linear–reverse-linear gap similarly decreases from 8.5 to 1.6. Thus, increasing pretraining weakens ordering effects on this suite without eliminating them.

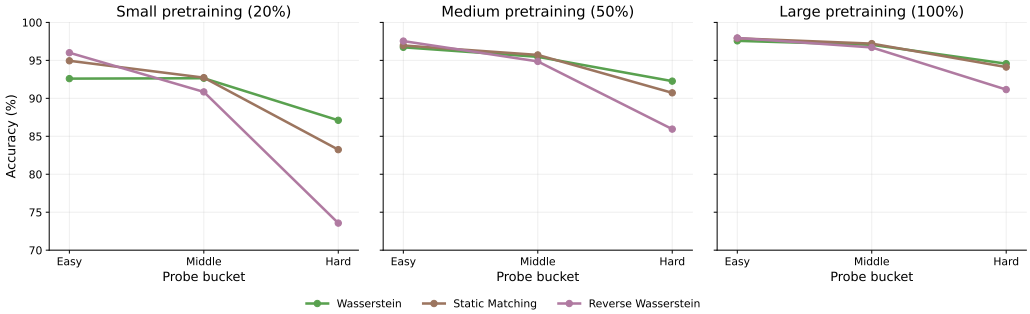


Figure 44: Ordering comparison within the Wasserstein family under a fixed small downstream budget. Panels vary pretraining amount. "Static Matching" denotes the exposure-matched static baseline.

Relation to pretrained SFT. The controlled synthetic analogue isolates pretraining as one factor behind the weaker curriculum effects observed in SFT. As pretraining increases, the differences among curricula shrink, static i.i.d. becomes stronger, and the easy-to-hard ordering advantage weakens. In the synthetic setting, these changes coincide with less remaining room for improvement, especially as easier levels approach saturation, suggesting one mechanism through which pretraining can reduce curriculum gains. Because saturation is less evident in the SFT experiments, other properties of natural post-training data may also matter.

Finding 9. *Pretraining weakens downstream curriculum effects, helping explain why they are weaker in pretrained SFT; less remaining room for improvement may be one mechanism.*