PROCEDURAL PRETRAINING FOR MOLECULAR PROPERTY PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular property prediction is often limited by the small size of labeled downstream datasets, motivating pretraining on large corpora of unlabeled molecules. In this work, we ask whether useful inductive biases can instead be learned from abstract, procedurally generated data before a model sees any molecular data. We introduce a three-stage training pipeline consisting of procedural pretraining, molecular pretraining on SMILES, and downstream fine-tuning, and evaluate several procedural tasks spanning sequence structure, cellular automata, and graph reasoning. We find that procedural pretraining can improve molecular property prediction even after subsequent molecular pretraining: on Lipophilicity, REVERSE reduces test error by 4.8%. For context, the magnitude of this improvement is roughly 90% of the performance difference between our 250K-molecule baseline and the publicly released MoLFormer checkpoint pretrained on approximately 100M molecules. Our analysis shows that the benefit is strongest under downstream data scarcity, depends on the structure of the procedural data rather than only surface-level statistics, and does not increase monotonically with additional procedural training. Instead, transfer typically peaks at an intermediate procedural budget and deteriorates as the model approaches convergence on the procedural task. We further find that, for several tasks, much of the transferable information is localized in the attention layers, while feed-forward layers can contribute to overspecialization. These results show that procedural data can provide transferable structure for molecular learning and offer a complementary route to improving performance when labeled molecular data are limited.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Modern molecular machine learning models are often trained in a regime where unlabeled molecules are abundant (Irwin & Shoichet, 2004; Kim et al., 2019) but labeled downstream data are scarce. Molecular property prediction datasets frequently contain only hundreds or thousands of labeled examples (Wu et al., 2018), making it difficult for high-capacity models to learn useful representations from downstream supervision alone. A common response has been to adopt the pretraining paradigm developed for language models: models are first trained on large collections of unlabeled molecular representations, such as SMILES strings, and subsequently fine-tuned for a target property (Chithrananda et al., 2020; Ross et al., 2022). While such molecular pretraining can substantially improve performance, acquiring and processing increasingly large molecular corpora

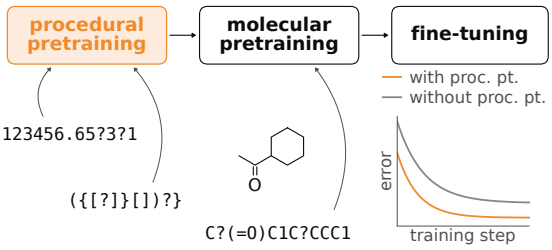


Figure 1: Overview of the proposed three-stage training pipeline. A transformer is first pretrained on procedurally generated masked-token tasks, then pretrained on molecular SMILES, and finally fine-tuned for molecular property prediction. Procedural pretraining can reduce downstream error relative to molecular pretraining and fine-tuning alone.

can be expensive, and it remains unclear whether useful inductive biases must themselves be learned from molecular data.

Recent work suggests an alternative source of pretraining signal: procedurally generated data. Simple algorithmic processes, such as logical operations (Wu et al., 2022; Shinnick et al., 2025), formal languages (Papadimitriou & Jurafsky, 2023; Hu et al., 2025), and cellular automata (Zhang et al., 2025; Lee et al., 2026), can generate effectively unlimited data with controlled structural properties. Procedural pretraining on such data has been shown to induce representations that transfer to natural language and vision tasks (Jiang et al., 2026; Shinnick et al., 2026). This raises a surprising possibility for molecular learning: *can a model learn useful computational structure before seeing a single molecule, and can that structure subsequently improve molecular property prediction?*

We investigate this question by introducing a procedural pretraining stage before conventional molecular pretraining and downstream fine-tuning. Using MoLFormer (Ross et al., 2022) as our base architecture, we pretrain on several procedural tasks spanning sequence manipulation (Jiang et al., 2026), hierarchical structure (Hu et al., 2025), dynamical systems (Lee et al., 2026), and graph reasoning, then transfer the learned transformer weights to molecular prediction. Our experiments consider both a realistic three-stage pipeline, procedural pretraining, molecular pretraining, and downstream fine-tuning, and a simplified setting that removes molecular pretraining to isolate the mechanisms responsible for transfer.

We find that procedural pretraining can improve molecular property prediction even after subsequent pretraining on real molecular data. On Lipophilicity (Wu et al., 2018), pretraining on a simple sequence-reversal task reduces test error by 4.8% relative to a molecular-pretraining-only baseline, recovering roughly 90% of the performance difference between our 250K-molecule baseline and the publicly released MoLFormer checkpoint pretrained on approximately 100M molecules. We then use controlled experiments to characterize when this transfer is beneficial, what properties of the procedural data drive it, and where the transferable information is represented in the model.

Our main contributions are:

- We show that **procedural pretraining can improve molecular property prediction even after subsequent molecular pretraining**, demonstrating that useful non-molecular structure can survive a realistic domain-specific pretraining stage.
- We find that **structural alignment** between procedural and molecular data provides an initial head start during molecular pretraining but **does not reliably predict downstream benefit**.
- We show that the benefit is **largest under downstream data scarcity**, suggesting that procedural pretraining acts as an inductive bias that is most valuable when labeled molecular data are limited.
- We find that **more procedural pretraining is not always better**. Across four of five tasks, downstream performance peaks at an intermediate procedural budget and deteriorates as the model approaches the optimum of the procedural task.
- We demonstrate that transfer depends on the **structure of the procedural data rather than surface-level statistics**: shuffling token order while preserving vocabulary, sequence length, and token frequencies removes the benefit.
- We **localize much of the transferable information to the attention layers** for several procedural tasks, while finding that feed-forward layers can contribute to the over-specialization observed at larger procedural budgets.

2 Related Work

Pretraining on procedural data. Recent work has shown that useful representations can be learned from procedurally generated data rather than natural corpora. In language, pretraining on artificial languages, formal languages, neural cellular automata, and simple algorithmic tasks can improve downstream performance on natural language tasks (Papadimitriou & Jurafsky, 2020; Hu et al., 2025; Lee et al., 2026; Jiang et al., 2026). Similar results have been observed in vision using synthetic images such as fractals or structured noise (Kataoka et al., 2021; Baradad Jurjo et al., 2021).

While many of these procedures reproduce structural properties of the target domain, recent work demonstrates transfer even across domains: sequence-based formal languages can improve vision transformers (Shinnick et al., 2026), and music pretraining can benefit language models (Nomura, 2026). This suggests that procedural pretraining can provide useful inductive biases without closely resembling downstream data.

Prior work has also investigated which properties make procedural pretraining transferable. Structural properties such as diversity, long-range dependencies, and hierarchical organization can improve transfer (Baradad Jurjo et al., 2021; Chiang & Lee, 2021; Hu et al., 2025). However, strong domain alignment is not always necessary: even simple sequence operations retain part of the benefit of natural-language pretraining (Wu et al., 2022), and transferable information may reside in different transformer components depending on the downstream task (Jiang et al., 2026). Together, these findings point to two possible sources of transfer: alignment between the procedural and target distributions, and more general changes to the model's inductive bias.

We extend this line of work to molecular modeling, where labeled data are comparatively scarce and procedural pretraining has received little attention. We study whether abstract procedural tasks can complement molecular pretraining, how alignment with SMILES structure affects transfer, and which model components and properties of procedural-task learning account for downstream gains.

3 METHODOLOGY

Training pipeline. We study a three-stage training pipeline consisting of (a) procedural pretraining, (b) molecular pretraining, and (c) downstream fine-tuning. The procedural stage exposes the model only to algorithmically generated data, before it encounters any molecular examples. We then optionally continue pretraining on unlabeled SMILES strings before fine-tuning on a molecular property prediction task. In Section 4.2, we omit the molecular pretraining stage for some experiments and fine-tune directly after procedural pretraining in order to isolate the effect of the procedural data.

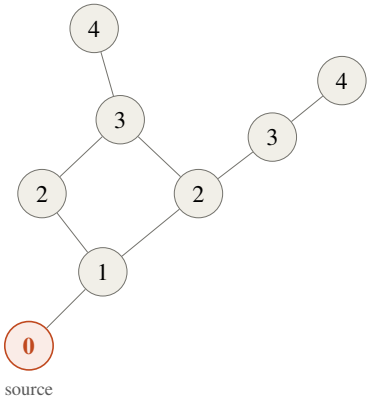
Procedural pretraining tasks. We consider five procedural tasks designed to expose the model to different forms of structure. Three operate directly on synthetic sequences: REVERSE, in which the model must recover the reversed form of a token sequence; DYCK, which consists of properly nested bracket sequences; and DYCK-SHUFFLE, which permits crossing bracket dependencies. DYCK mirrors the structure of branching parentheses in SMILES, while DYCK-SHUFFLE mirrors ring-closure digits. We additionally consider neural cellular automaton (NCA) trajectories, which introduce spatiotemporal structure, and a shortest-path (SSP) task in which the model predicts node distances from the SMILES representation of a random graph. All tasks are formulated as masked-token prediction problems so that the same model architecture and training objective can be used throughout procedural and molecular pretraining. Examples of the procedural inputs are shown in Figure 2.

Model. We use MoLFormer (Ross et al., 2022), an encoder-only transformer with approximately 44M parameters designed for learning from SMILES representations. MoLFormer uses rotary positional embeddings (RoPE), linear attention based on generalized random features, and a regular-expression-based SMILES tokenizer with a vocabulary of 2,362 tokens. We use the same transformer architecture across all training stages.

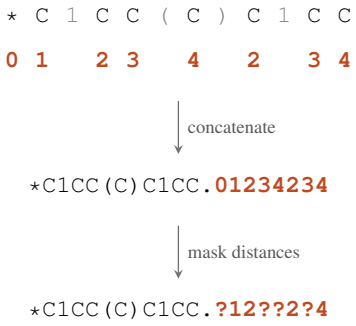
Molecular pretraining. MoLFormer is conventionally pretrained using masked token modeling (MTM) on large collections of unlabeled SMILES strings drawn from PubChem (Kim et al., 2019) and ZINC (Irwin & Shoichet, 2004). The publicly released MoLFormer checkpoint was trained on approximately 100M molecules, which is beyond our available compute budget. To evaluate procedural pretraining within a realistic molecular pretraining pipeline, we therefore construct a smaller-scale approximation of the setup in Ross et al. (2022). We pretrain for 12 epochs on a 250k-molecule subset of PubChem using the same MTM objective and a masking probability of 0.15. When comparing procedural and non-procedural initializations, all models receive the same molecular pretraining data and compute budget.

Downstream evaluation. We evaluate on three molecular property prediction benchmarks with continuous regression targets: Lipophilicity from MoleculeNet (Wu et al., 2018), containing ap-

Task	Generated	After masking
Reverse	`BCAFED.DEFACB`	`BCAFED.?EF??B`
Dyck	`()()[]{[()]}`	`()(?[?{[(?]}`
Dyck-Shuffle	`([)]{}(){[}]`	`([?]{?(){[}?`



(a) graph and source node



(b) input sequence

Figure 2: Procedural pretraining tasks. Top: example sequences of the sequence-level tasks before and after masking. Bottom: how an input sequence for the SSP task is created. a) A molecule-like random graph is generated and a source node chosen at random. b) The SMILES representation of the graph (using $\star$ for the source and `C` for all other nodes) is concatenated with shortest-path distances from the source to all other nodes.

proximately 4k molecules; FreeSolv, containing 642 molecules; and QM9, for which we predict the HOMO–LUMO gap and use either the full dataset or controlled subsets of the training set. Models are fine-tuned for 500 epochs. We report test mean absolute error (MAE) on standardized targets from the checkpoint achieving the best validation performance, following Ross et al. (2022). Unless stated otherwise, comparisons use identical downstream splits and optimization settings.

Weight transfer. We partition the model parameters into token embeddings, attention projections, feed-forward layers, LayerNorm parameters, and the output head. The procedural and molecular domains use different token semantics, so we do not transfer learned procedural token embeddings. Instead, the embedding matrix is held fixed at its random initialization during procedural pretraining, ensuring that any transferable information must be encoded in the transformer backbone rather than in task-specific token representations. We also discard the procedural output head, whose dimensionality and semantics need not match those of subsequent stages. Our default transfer setting therefore retains the learned transformer backbone while reinitializing the input and output interfaces as required.

To localize where transferable information is stored, we additionally consider selective transfer. We transfer either only the attention-layer weights or only the feed-forward-layer weights, while reinitializing all remaining parameters, including LayerNorms. This allows us to measure how much of the downstream benefit can be attributed to each component of the transformer.

4 Experiments

4.1 Procedural Pretraining Improves Molecular Property Prediction

We first evaluate procedural pretraining in the full three-stage pipeline: procedural pretraining, molecular pretraining, and downstream fine-tuning. We compare models initialized from each procedural task against a baseline trained only on molecular data, while keeping the molecular pretraining and downstream fine-tuning stages fixed. We evaluate on Lipophilicity, FreeSolv, and a 1k-molecule subset of QM9 for HOMO–LUMO gap prediction.

Table 1: Molecular property prediction after procedural and molecular pretraining: All models are pretrained on our 250k molecule PubChem subset before downstream fine-tuning. Standardized test MAE is reported (lower is better), with Δ relative to the molecular-pretraining-only baseline. Results average n=3 seeds (n=6 for FreeSolv); best procedural results are bold.

	LIPOPHILICITY		QM9 GAP (1k)		FREESOLV	
	MAE	Δ%	MAE	Δ%	MAE	Δ%
Random init. → PUBCHEM250K (baseline)	0.4083 ±0.0125	—	0.2662 ±0.0015	—	0.1676 ±0.0145	—
Procedural pretraining → PUBCHEM250K						
DYCK	0.4028 ±0.0020	−1.29	0.2641 ±0.0020	−0.80	0.1764 ±0.0163	+5.69
DYCK-SHUFFLE	0.3937 ±0.0029	−3.53	0.2680 ±0.0026	+0.65	0.1867 ±0.0129	+12.36
REVERSE	**0.3885** ±0.0111	**−4.80**	**0.2611** ±0.0004	**−1.91**	**0.1621** ±0.0031	**−2.59**
NCA	0.4069 ±0.0027	−0.28	0.2703 ±0.0054	+1.51	0.1849 ±0.0110	+11.16
SMILES SHORTEST-PATHS	0.4011 ±0.0016	−1.70	0.2671 ±0.0001	+0.32	0.1713 ±0.0074	+3.13

Table 1 reports standardized test MAE and the relative change with respect to the molecular-pretraining-only baseline. On Lipophilicity, two procedural tasks yield clear improvements: REVERSE reduces MAE by 4.80% and DYCK-SHUFFLE by 3.53%, with both improvements separated from the baseline across seeds. The remaining three tasks also improve the mean performance, although their gains lie within seed variation. REVERSE additionally improves performance on the 1k-molecule QM9 subset by 1.91%.

The magnitude of the Lipophilicity improvement is notable relative to scaling molecular pretraining itself. Fine-tuning the publicly released MoLFormer checkpoint, pretrained on approximately 100M molecules, improves over our 250k-molecule molecular-pretraining baseline by 5.35%. By comparison, only 1.5k procedural pretraining steps on REVERSE recover a 4.80% improvement without using any additional molecular data.

We do not observe a clear benefit on FreeSolv. Four procedural tasks perform worse than the molecular-pretraining-only baseline, although the released MoLFormer checkpoint, trained on approximately $400\times$ more molecular data, is also worse by 12.01%. Given that FreeSolv contains only 642 molecules, the dataset appears too small to reliably resolve differences of this magnitude.

Overall, these results show that information learned from procedural data can survive subsequent molecular pretraining and improve downstream molecular property prediction.

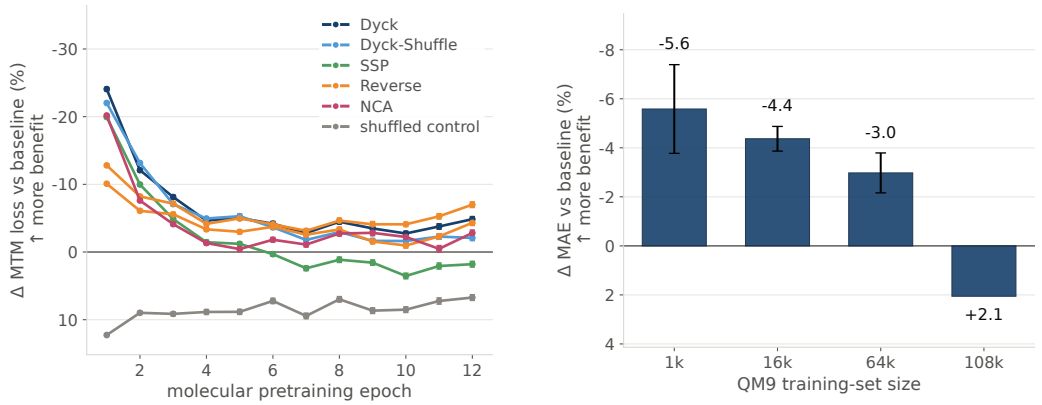
4.2 WHEN AND WHY DOES PROCEDURAL PRETRAINING HELP?

Having established that procedural pretraining can improve the full three-stage molecular learning pipeline, we next investigate when and why this transfer occurs. Unless otherwise noted, experiments in this section use a two-stage diagnostic setting: procedural pretraining followed directly by downstream fine-tuning, with no intervening molecular pretraining. In this setting, procedurally pretrained models are compared against an identically fine-tuned, randomly initialized baseline. Removing the molecular stage isolates the effects of procedural pretraining from representations subsequently learned from SMILES data. Two subsections also examine the full pipeline: Section 4.2.1 retains molecular pretraining throughout, while Section 4.2.4 reports shuffled-token controls in both settings.

4.2.1 DOES STRUCTURAL ALIGNMENT WITH MOLECULES MATTER?

REVERSE provides the strongest downstream improvement in Section 4.1, despite having no obvious structural correspondence with SMILES. By contrast, DYCK, DYCK-SHUFFLE, and SSP were chosen to reflect structural properties that are more closely aligned with molecular representations. We therefore ask whether such alignment provides an advantage during molecular pretraining.

We compare held-out masked-token modeling loss throughout 12 epochs of molecular pretraining on PubChem250k. As shown in Figure 3a, the SMILES-aligned tasks, DYCK, DYCK-SHUFFLE, and SSP, provide the largest advantage at the beginning of molecular pretraining. However, this advantage decays rapidly, and after several epochs their performance becomes similar to that of REVERSE. SSP eventually becomes detrimental despite operating on real SMILES strings.



(a) Held-out PubChem250k loss.

(b) QM9 gap MAE across training-set sizes.

Figure 3: Molecular pretraining dynamics and data-scarcity evaluation: a) Held-out molecular pretraining loss for each procedural task at its best configuration; b) QM9 gap MAE across training-set sizes after DYCK ($k = 64$) pretraining. Values are relative to the randomly initialized baseline; error bars are ± 1 standard deviation over n=3 seeds. The full-dataset point in b) is a single seed. Panel (a) also includes the token-shuffled DYCK-SHUFFLE control discussed in Section 4.2.4

These results suggest that structural alignment provides an initial head start during molecular pretraining, but that much of this advantage can be recovered from SMILES data alone. More broadly, structural alignment is not a reliable predictor of which tasks will transfer best: REVERSE provides the strongest downstream improvement after the full pipeline despite having little obvious alignment with SMILES. Thus, within our task suite, procedural tasks do not need to be structurally aligned with the molecular target domain to transfer effectively.

4.2.2 THE BENEFIT IS LARGEST UNDER DATA SCARCITY

If procedural pretraining supplies useful structure that would otherwise have to be inferred from downstream molecular data, its value should increase as the amount of labeled data decreases. To test this hypothesis, we subsample the QM9 training set to 1k, 16k, 64k, and the full 108,446 molecules. At each size, we compare a model pretrained on DYCK against a randomly initialized baseline and evaluate both models on the same held-out test set of 13,389 molecules.

Figure 3b shows a clear relationship between downstream dataset size and transfer benefit: the advantage of procedural pretraining grows monotonically as the amount of training data decreases. Procedural pretraining therefore acts most strongly as an inductive bias in low-data regimes, where the downstream dataset provides less opportunity for the model to learn the relevant structure directly.

4.2.3 MORE PROCEDURAL PRETRAINING IS NOT ALWAYS BETTER

A practical appeal of procedural data is that it can be generated in effectively unlimited quantities. We therefore ask whether increasing the procedural pretraining budget continually improves downstream transfer.

We vary both task complexity and procedural training duration. For DYCK and DYCK-SHUFFLE, complexity is controlled by the number of distinct bracket types; for REVERSE, by sequence length; and for NCA, by the gzip complexity of the generated trajectories. For each combination of task complexity and training budget, we procedurally pretrain a model and compare its downstream Lipophilicity performance against the randomly initialized baseline.

As shown in Figure 4, four of the five procedural tasks exhibit the same qualitative pattern: downstream benefit initially increases with additional procedural training, reaches a maximum at an intermediate budget, and then deteriorates. At the largest budgets, performance is often worse than the randomly initialized baseline. NCA is the only exception and is also the only task that the model does not learn to solve during procedural pretraining.

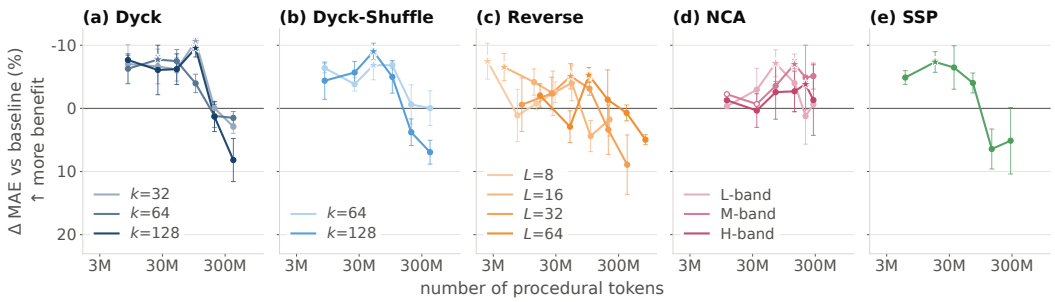


Figure 4: Downstream transfer across procedural pretraining budgets: Lipophilicity performance relative to random initialization across procedural budgets and task-complexity settings. Error bars show ± 1 standard deviation over $n=3$ seeds; across four of five tasks, downstream benefit vanishes at higher budgets. In particular, a few seeds at the highest budget collapsed to the loss of a constant predictor. These seeds are excluded from the mean calculation.

This behavior suggests that transfer degrades as the model becomes increasingly specialized to the procedural objective. To examine this more directly, we compare downstream benefit with the procedural loss reached at each training budget. For deterministic tasks such as DYCK, REVERSE, and SSP, the Bayes-optimal loss is zero, and the models approach this value at large training budgets. DYCK-SHUFFLE is not perfectly predictable because closing-bracket identity contains aleatoric uncertainty: at a masked position, multiple currently open bracket types can yield a valid sequence. Its Bayes-optimal loss is therefore non-zero and can be computed from the generating process. For $k = 128$, this value is approximately 1.267, which the model approaches at the largest budgets. Procedural losses and Bayes-optima can be found in Appendix C.

The relevant factor is therefore not low procedural loss in absolute terms, but proximity to the best achievable solution for the procedural task. Across tasks, downstream transfer is strongest before the model approaches procedural convergence, suggesting that useful general structure is acquired earlier than task-specific specialization.

4.2.4 TRANSFER REQUIRES PROCEDURAL STRUCTURE

One alternative explanation for the observed gains is that procedural pretraining need not teach meaningful structure at all. Any additional pretraining could move the model away from its random initialization, calibrate parameter scales, or expose it to relevant sequence lengths and token-frequency statistics. To distinguish these effects from learning structured dependencies, we construct shuffled-token controls. We randomly permute token order within each procedural sequence, thereby destroying order-dependent structure while preserving the vocabulary, sequence lengths, and token frequencies of the original corpus.

We evaluate these controls in two complementary settings. First, in the two-stage diagnostic setting used throughout most of Section 4.2, we construct a separate shuffled-token corpus for each of the five procedural tasks. Each model is pretrained on the shuffled corpus and then fine-tuned directly on Lipophilicity, without molecular pretraining. As shown in Appendix D, every shuffled-token model loses the downstream benefit of its structured counterpart and performs no better than the randomly initialized fine-tuning baseline.

Second, we test whether this conclusion survives the full three-stage pipeline using DYCK-SHUFFLE as a representative control. The token-shuffled model undergoes the same molecular pretraining and downstream fine-tuning as the structured model. In Figure 3a, its held-out MTM loss remains worse than that of molecular pretraining from random initialization throughout all 12 epochs. After Lipophilicity fine-tuning, it obtains an MAE of 0.4181 ± 0.0062, a roughly 2.46% degradation relative to the molecular-pretraining-only baseline, whereas structured DYCK-SHUFFLE improves performance by 3.53%.

These results rule out generic exposure to synthetic sequences or their token-level statistics as a sufficient explanation for transfer. The downstream benefit depends on the structured relationships present in the procedural data.

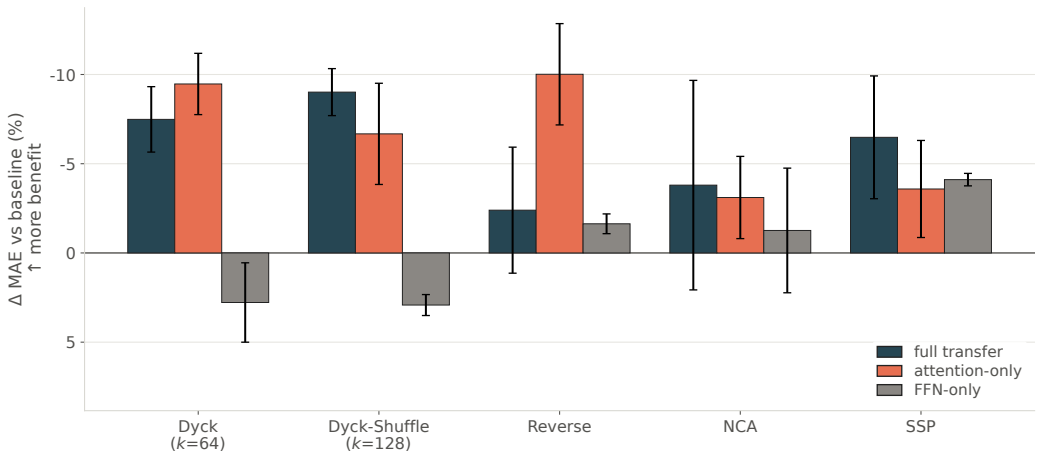


Figure 5: Lipophilicity benefit for full-backbone, attention-only, and feed-forward-only transfer. Attention-only transfer retains most of the benefit, while feed-forward-only transfer generally does not.

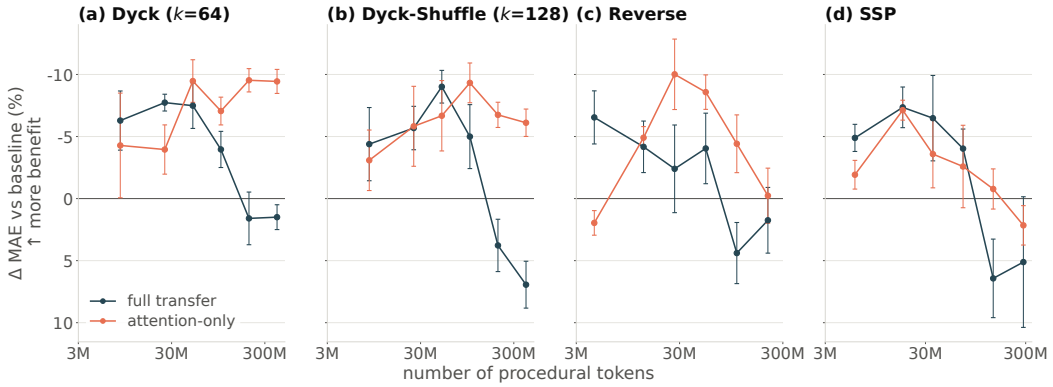


Figure 6: Full-backbone transfer compared with attention-only transfer at increasing procedural token budgets. We report Lipophilicity benefit over the randomly initialized baseline. Error bars are ± 1 standard deviation over $n = 3$ seeds. Attention-only transfer decays more slowly than full transfer; DYCK and DYCK-SHUFFLE retain a benefit at the largest budgets, while REVERSE and SSP still converge towards the baseline.

4.3 WHERE DOES THE TRANSFER RESIDE?

The preceding experiments show that procedural structure can transfer to molecular prediction, but they do not identify where that information is represented within the transformer. We therefore ask which model components carry the transferable information learned during procedural pretraining.

We compare three transfer settings on Lipophilicity: transferring the full transformer backbone, transferring only the attention-layer weights, and transferring only the feed-forward-layer weights. All other parameters are reinitialized as described in Section 3.

Figure 5 shows that attention-only transfer matches or exceeds full-backbone transfer for DYCK, DYCK-SHUFFLE, and REVERSE. In contrast, transferring only the feed-forward layers removes most of the downstream benefit and can even hurt performance. SSP behaves differently: neither component alone recovers the full-transfer benefit, suggesting that useful information is distributed across both attention and feed-forward layers. NCA provides no significant benefit under any transfer configuration.

The deterioration observed at large procedural budgets in Section 4.2.3 raises a related question: does this over-specialization also reside in particular model components? To test this, we repeat the procedural-budget sweep while transferring only the attention-layer weights to the downstream task.

Figure 6 shows that attention-only transfer is generally more robust to increasing procedural budgets than full-backbone transfer. For DYCK and DYCK-SHUFFLE, the downstream benefit no longer decays at the largest tested budgets when only attention weights are transferred. REVERSE and SSP also decay more slowly under attention-only transfer, although both eventually converge back towards the baseline.

Together, these results indicate that much of the useful transferable structure learned from DYCK, DYCK-SHUFFLE, and REVERSE is carried by the attention layers. They also suggest that part of the degradation caused by excessive procedural pretraining is associated with the feed-forward layers: discarding these weights preserves useful transfer over a wider range of procedural budgets. The separation is not complete, however, since REVERSE and SSP still lose their advantage at sufficiently large budgets even under attention-only transfer.

5 CONCLUSION AND OPEN QUESTIONS

We showed that procedural pretraining on abstract, non-molecular data can improve molecular property prediction, even after subsequent pretraining on real molecular data. The gains are largest in low-data regimes, and in our strongest result, a simple sequence-reversal task recovers roughly 90% of the performance difference between our 250K-molecule baseline and the publicly released MoLFormer checkpoint pretrained on approximately 100M molecules. The benefit depends on the structure of the procedural data rather than on simple token-level statistics: shuffling token order removes the transfer advantage.

Our results also show that more procedural pretraining is not always better. Across most tasks, downstream performance peaks at an intermediate training budget and deteriorates as the model approaches procedural convergence, suggesting a transition from learning broadly useful structure to task-specific specialization. For several tasks, much of the transferable information is retained in the attention layers, while the feed-forward layers appear to contribute more strongly to over-specialization.

Several questions remain open. What properties make a procedural task transferable, and can they be identified without downstream evaluation? Can the optimal stopping point for procedural pretraining be predicted from procedural training dynamics alone, rather than from downstream validation? Our results were also inconclusive on FreeSolv and for the NCA task, which showed no clear downstream benefit at any procedural budget. Determining whether these outcomes reflect dataset size, task predictability, or a more fundamental mismatch remains an important direction for future work. More broadly, do these effects persist across larger models, alternative molecular representations, and other scientific domains? Our findings suggest that improving molecular learning may require not only more molecular data, but also better ways of shaping the model before it sees a single molecule.

AI USE STATEMENT

In this work, we used generative AI tools to write parts of the implementation. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning or reformatting datasets, or interpreting results. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in writing proofs, assisting with translation and supporting qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI to polish figures and writing, but not to draft entire paragraphs. Generative AI tools were used to support literature search and identification of related work, complementing manual search. The authors reviewed all AI-generated text and code and verified suggested references against the original papers. We take responsibility for the final content of this work, including text, claims, figures, code and results.

REFERENCES

Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to See by Looking at Noise. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2556–2569. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/14f2ebeab937ca128186e7ba876faef9-Abstract.html.

Cheng-Han Chiang and Hung-yi Lee. On the Transferability of Pre-trained Language Models: A Study from Artificial Datasets, September 2021. URL https://arxiv.org/abs/2109.03537v2.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction, October 2020. URL http://arxiv.org/abs/2010.09885. arXiv:2010.09885 [cs.LG].

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between Circuits and Chomsky: Pre-pretraining on Formal Languages Imparts Linguistic Biases, May 2025. URL http://arxiv.org/abs/2502.19249. arXiv:2502.19249 [cs.CL].

John J. Irwin and Brian K. Shoichet. ZINC - A Free Database of Commercially Available Compounds for Virtual Screening. *Journal of Chemical Information and Modeling*, 45(1):177–182, December 2004. ISSN 1549-9596. doi: 10.1021/ci049714+. URL https://doi.org/10.1021/ci049714+.

Liangze Jiang, Zachary Shinnick, Anton van den Hengel, Hemanth Saratchandran, and Damien Teney. Procedural pretraining: Warming up language models with abstract data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without Natural Images, January 2021. URL http://arxiv.org/abs/2101.08515. arXiv:2101.08515 [cs.CV].

Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, January 2019. ISSN 0305-1048. doi: 10.1093/nar/gky1033. URL https://doi.org/10.1093/nar/gky1033.

Dan Lee, Seungwook Han, Akarsh Kumar, and Pulkit Agrawal. Training Language Models via Neural Cellular Automata, March 2026. URL http://arxiv.org/abs/2603.10055. arXiv:2603.10055 [cs.LG].

Yoshinori Nomura. Listen and Chant Before You Read: The Ladder of Beauty in LM Pre-Training, April 2026. URL http://arxiv.org/abs/2604.21265. arXiv:2604.21265 [cs.CL].

Isabel Papadimitriou and Dan Jurafsky. Learning Music Helps You Read: Using Transfer to Study Linguistic Structure in Language Models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6829–6839, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.554. URL https://aclanthology.org/2020.emnlp-main.554/.

Isabel Papadimitriou and Dan Jurafsky. Injecting structural hints: Using language models to study inductive biases in language learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-Scale Chemical Language Representations Capture Molecular Structure and Properties, December 2022. URL http://arxiv.org/abs/2106.09553. arXiv:2106.09553 [cs.LG].

Zachary Shinnick, Liangze Jiang, Hemanth Saratchandran, Anton van den Hengel, and Damien Teney. Transformers pretrained on procedural data contain modular structures for algorithmic reasoning. *arXiv preprint arXiv:2505.22308*, 2025.

Zachary Shinnick, Liangze Jiang, Hemanth Saratchandran, Damien Teney, and Anton van den Hengel. Can you learn to see without images? procedural warm-up for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Yuhuai Wu, Felix Li, and Percy Liang. Insights into Pre-training via Simpler Synthetic Tasks. October 2022. URL https://openreview.net/forum?id=XiLasGufCM.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A Benchmark for Molecular Machine Learning, October 2018. URL http://arxiv.org/abs/1703.00564. arXiv:1703.00564 [cs.LG].

Shiyang Zhang, Aakash Patel, Syed A Rizvi, Nianchen Liu, Sizhuang He, Amin Karbasi, Emanuele Zappala, and David van Dijk. Intelligence at the edge of chaos, 2025. URL https://arxiv.org/abs/2410.02536.

A Reproducibility and Training Details

A.1 Training details

All three stages use the same MoLFormer encoder (Ross et al., 2022): 12 layers, hidden size 768, 12 heads of dimension 64, feed-forward width 768, GELU activations, linear attention with 32 generalized random features, rotary position embeddings, a maximum length of 202 tokens and the released 2,362-token SMILES vocabulary (46.8M parameters with the MTM head, 48.0M with the regression head). All runs use one GPU (NVIDIA RTX 6000 24GB or A40 48GB), PyTorch 1.7.1 with PyTorch Lightning 1.1.5 and apex 22.03; procedural corpora are generated with numpy, RDKit and networkx (SSP) and the JAX generator of Lee et al. (2026) (NCA).

Table 2 summarizes the training configuration for each stage. All stages use `apex` FusedLAMB with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-6}$, no weight decay, and global gradient-norm clipping at 1.0. The molecular-pretraining and fine-tuning configurations follow the released MoLFormer implementation (Ross et al., 2022), with three exceptions: for molecular pretraining we reduce the batch size from 1,200 to 256 and add a 500-step linear warm-up, and we fine-tune on QM9 for 400 rather than 500 epochs.

Token embeddings remain fixed at their random initialization during procedural pretraining and are trainable during molecular pretraining and downstream fine-tuning. Fine-tuning targets are standardized using statistics computed from the training split, and the checkpoint with the lowest validation MAE is used for test evaluation. Each procedural and each molecular pretraining is a single run with seed 12345. Fine-tuning uses seeds 12345, 42 and 43 (FreeSolv additionally uses 44, 45 and 46). The seed also draws the QM9 training subset.

Parameter	Procedural pretraining	Molecular pretraining	Downstream fine-tuning
Objective	MTM with task-specific masking	MTM with 15% of tokens selected for prediction	Regression from the mean-pooled encoder output; ℓ_1 loss on standardized targets
Peak learning rate	2.4×10^{-4}	2.4×10^{-4}	3×10^{-5}
Learning-rate schedule	10% linear warm-up, followed by cosine decay to 10% of the peak rate	500-step linear warm-up, followed by a constant rate	Constant; no warm-up
Batch size	256 sequences	256 molecules	128 molecules
Dropout probability	0.2	0.2	0.1
Training budget	500–24,000 steps	12 epochs	500 epochs for Lipophilicity and FreeSolv; 400 for QM9

Table 2: Training configurations for the three stages of the experimental pipeline.

A.2 Procedural-task generation

Table 3 summarizes the construction and masking policy of each procedural task. The values evaluated for the task complexity parameters are reported in Figure 7. For Reverse and SSP, sequences are processed using the MoLFormer tokenizer and masked as in molecular pretraining: of the positions selected for prediction, 80% are replaced by `[MASK]`, 10% by a random token, and 10% remain unchanged. Dyck, Dyck-Shuffle, and NCA are represented directly as integer token IDs, and every selected position is replaced by `[MASK]`.

For each task configuration and training budget of B steps, we generate $B \times 256$ sequences once, using random seed 42, and train for a single pass over them; the checkpoint at step B, the end of the cosine schedule, is the one transferred, and no validation set is held out.

Task	Generated sequence	Complexity parameter	Masking strategy
REVERSE	A sequence of L symbols drawn from an alphabet of 13 SMILES atom tokens, followed by `.` and the same symbols in reverse order	Sequence length L	Reversed tokens ($p_{\text{mask}} = 0.5$)
DYCK	A 64-token balanced bracket string with k distinct bracket types. An opening bracket is sampled with probability 0.5 whenever permitted to form a valid balanced-bracket string	Number of distinct bracket types k	Closing brackets ($p_{\text{mask}} = 0.5$)
DYCK-SHUFFLE	Similar to DYCK, except that a closing bracket may match any currently open bracket of the same type, allowing bracket pairs to cross	Number of distinct bracket types k	Closing brackets ($p_{\text{mask}} = 0.5$)
SSP	The canonical SMILES representation of a random connected graph with 8–25 nodes and node degrees 1–4, followed by `.` and, for every node in SMILES order, its shortest-path distance to a source node marked as $\star$	–	Distance tokens ($p_{\text{mask}} = 0.5$)
NCA	Ten frames of an 8×8 cellular automaton with six cell states, sampled two update steps apart. Each frame is represented by 16 tokens corresponding to non-overlapping 2×2 patches	gzip-compressibility band of the update rule	Uniformly across all patch tokens ($p_{\text{mask}} = 0.15$)

Table 3: Procedural-task generation and masking. For each task, we report the generated sequence, the complexity parameter varied in our experiments, and the tokens eligible for masking. Each eligible token is selected independently with probability p_{mask}.

B CHOOSING TASK COMPLEXITY AND PROCEDURAL BUDGET

Figure 7 shows the validation grid used to select task complexity and procedural-pretraining budget for each task.

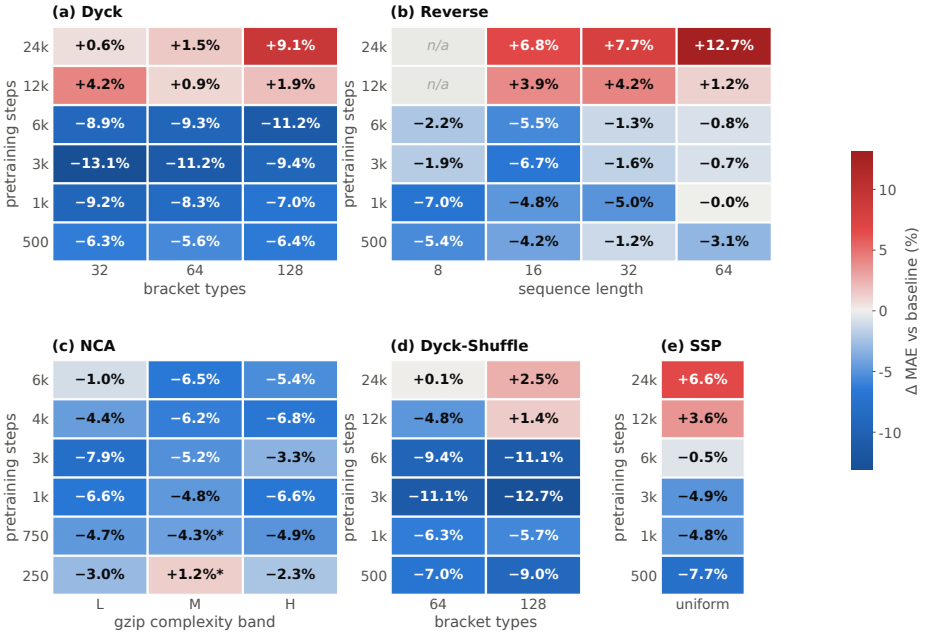


Figure 7: Grid search over task complexity and procedural-pretraining budget. Cells report mean validation-set Lipophilicity benefit over the randomly initialized baseline across $n = 3$ seeds.

C Procedural Pretraining Losses and Bayes-Floor Estimates

In Table 4, we report the procedural loss reached at every budget and the downstream benefit of the respective model on Lipophilicity over the random initialization baseline. We also mention the Bayes-optimal achievable loss for each task setting where we are able to determine it.

Task	Metric	Pretraining budget b_1	b_2	b_3	b_4	b_5	b_6	$\mathcal{L}^\star$
Dyck								
k=32	Δ MAE (%)	-7.0	-6.6	-6.2	-10.6	-0.0	$+2.9$	
	procedural loss	1.46	0.72	0.266	0.0512	6.76×10^{-3}	5.62×10^{-4}	0
k=64	Δ MAE (%)	-6.3	-7.7	-7.5	-4.0	$+1.6$	$+1.5$	
	procedural loss	1.98	1.21	0.527	0.158	0.0144	8.73×10^{-4}	0
k=128	Δ MAE (%)	-7.7	-6.1	-6.2	-9.6	$+1.3$	$+18.2$	
	procedural loss	1.86	1.11	0.47	0.0943	7×10^{-3}	6.81×10^{-4}	0
Dyck-Shuffle								
k=64	Δ MAE (%)	-6.4	-3.8	-6.8	-6.8	-0.6	-0.0	
	procedural loss	2.12	1.58	1.44	1.34	1.27	1.26	1.250
k=128	Δ MAE (%)	-4.4	-5.7	-9.0	-5.0	$+3.8$	$+6.9$	
	procedural loss	2.36	1.82	1.57	1.41	1.31	1.28	1.267
Reverse								
L=8	Δ MAE (%)	-7.5	$+1.1$	-0.8	-2.6	–	–	
	procedural loss	0.907	6.65×10^{-4}	4.39×10^{-6}	1.16×10^{-5}	–	–	0
L=16	Δ MAE (%)	-6.5	-4.2	-2.4	-4.0	$+4.4$	$+1.7$	
	procedural loss	1.65	0.229	1.07×10^{-4}	1.37×10^{-5}	6.44×10^{-7}	1.28×10^{-7}	0
L=32	Δ MAE (%)	-0.6	-2.4	-5.1	-3.1	$+3.4$	$+8.9$	
	procedural loss	2.03	1.27	0.302	1.68×10^{-4}	4.77×10^{-5}	2×10^{-6}	0
L=64	Δ MAE (%)	-2.0	$+2.9$	-5.3	-1.4	$+0.7$	$+5.0$	
	procedural loss	2.27	2.07	1.73	1.25	0.567	0.0474	0
SSP								
uni	Δ MAE (%)	-4.9	-7.4	-6.5	-4.0	$+6.4$	$+5.1$	
	procedural loss	0.894	0.691	0.592	0.463	0.176	0.0622	0
NCA								
band L	Δ MAE (%)	-0.3	-3.0	-7.1	-4.0	$+1.2$	-0.6	
	procedural loss	2.47	2.28	2.13	1.86	1.78	1.73	unk.
band M	Δ MAE (%)	-2.2	-0.7	-3.6	-7.0	-4.9	-5.1	
	procedural loss	3.75	3.58	3.46	3.41	3.36	3.27	unk.
band H	Δ MAE (%)	-1.3	$+0.3$	-2.6	-2.7	-3.8	-1.3	
	procedural loss	5.52	5.34	5.26	5.22	5.19	5.19	unk.

Table 4: Procedural loss and downstream benefit for each tested task-complexity-duration combination. $L^\star$ denotes the Bayes-optimal loss, which is unknown in closed form for NCA. Budgets b_1–b_6 are $\{500, 1500, 3000, 6000, 12000, 24000\}$ optimizer steps for the sequence tasks and $\{250, 750, 1500, 3000, 4500, 6000\}$ for NCA.

D SHUFFLING TOKEN ORDER REMOVES DOWNSTREAM BENEFIT ACROSS ALL PROCEDURAL TASKS

In Section 4.2.4, we found that shuffling the token order in the procedural sequences eliminates all downstream benefit. This section reports the shuffled-token ablation in the two-stage diagnostic setting for all five procedural tasks: procedural pretraining is directly followed by Lipophilicity fine-tuning. Figure 8 shows that every procedural task loses its downstream benefit when the token order within the sequences is shuffled.

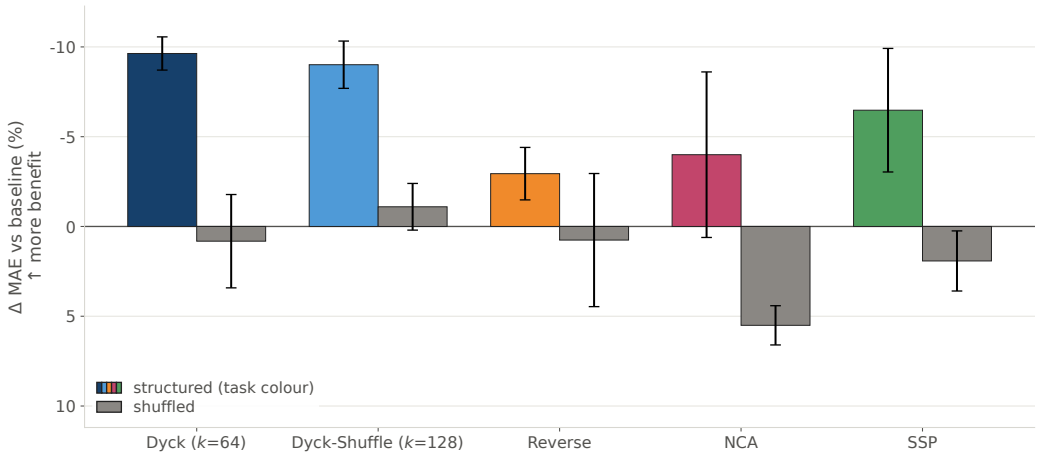


Figure 8: Token shuffling removes the downstream benefit of procedural pretraining across all five tasks in the two-stage diagnostic setting. Models are pretrained on either the structured corpus or a version in which tokens are shuffled within each sequence, and are then fine-tuned directly on Lipophilicity without molecular pretraining. Shuffling preserves each sequence's length and token counts while disrupting order-dependent structure, indicating that marginal token statistics alone are insufficient to explain the observed benefit.