Towards Understanding Self-Pretraining for Sequence Classification

Omar Coser [1 2] **Antonio Orvieto** [3 4 5]

Abstract

It was recently shown by Amos et al. (2023) that to boost test accuracy of transformer models on sequence classification, it can be highly effective to first pretrain with a masked token prediction objective on exactly the same data (self-pretraining, SPT). While the focus of Amos et al. (2023) is to show that transformers – and not only state-space models (SSMs, like S4) – can perform well on the Long-Range Arena (LRA, a collection of challenging synthetic sequence classification tasks), their finding is intriguing from a more fundamental perspective. Indeed, even though it can be easily claimed that the observed gains come from the benefits of data-driven initialization and pretraining inductive biases, it is unclear which precise mechanism unlocks performance and why standard supervised learning can fail. To better understand this intriguing phenomenon, we replicate and ablate the results of Amos et al. (2023). We show that substantial gains can be observed even at an extremely small scale, using a self-pretraining pipeline that requires little extra compute. We further identify in the attention mechanism weights the source of SPT improved performance. We hope our insights lead to future investigations around SPT, and that our work exposes this unusual yet promising technique for data-scarce learning to a broader audience.

1. Introduction

The Long-range Arena (Tay et al., 2020b) has served great importance for the development of new efficient token-mixing strategies over the last few years. A prime example is that of the first state-space-model (SSM): S4 (Gu et al., 2022b), that became popular for surpassing transformers on the LRA with 20% average increase in accuracy. S4 along with later variants also developed by benchmarking on the LRA (Smith et al., 2023; Gu et al., 2022a; Poli et al., 2023; Orvieto et al., 2023) serve as the basis for modern architectures such as Mamba-1/2 (Gu & Dao, 2024; Dao & Gu, 2024), GLA (Yang et al., 2024a), RetNet (Sun et al., 2023), RWKV (Peng et al., 2023; 2024) and DeltaNet (Yang et al., 2024b) among others.

While the latest developments in token-mixing mechanisms are mainly inspired by language modeling performance (Waleffe et al., 2024), researchers maintained a strong interest in explaining the gap between transformers and SSMs on LRA (e.g. Zimerman & Wolf (2023)). In their brilliant contribution, awarded the *Outstanding Paper Award at ICLR 2024*, Amos et al. (2023) showed that training transformer models from scratch (as done in the LRA benchmark) leads to an under-estimation of their performance and demonstrates that dramatic gains can be achieved with a pretraining $\rightarrow$ finetuning setup (see Table 1). At a first glance, this finding may not seem too surprising – yet, **pretraining is here performed on the same data** (self-pretraining, **SPT**), that is: the transformer is first tasked to learn to predict masked tokens (or the next token) in the sequence, and only later is trained on tasks labels. While this procedure differs substantially with the usual pretrainig paradigm (large pretraining corpus, small finetuning dataset, e.g. Brown et al. (2020)), it draws conceptual similarities both with classical curriculum learning (Bengio et al., 2009) and more recent applied research: among others, El-Nouby et al. (2021) showed the efficacy of self-pretraining in vision and more recently Krishna et al. (2023) showed that pretraining large language models directly on downstream datasets could often match pretraining on massive external data, highlighting that much of the observed *performance gains may be driven by the pretraining objective itself rather than the data volume*.

Compared to El-Nouby et al. (2021); Krishna et al. (2023), the main objective of Amos et al. (2023) was not to show how to reach state-of-the-art performance or to eliminate the need of pretaining on a massive external dataset. Instead, they show that on challenging tasks such as the LRA,

[*]Equal contribution [1]Unit of Artificial Intelligence and Computer Systems, Università Campus Bio-Medico, Rome, Italy [2]Unit of Advanced Robotics and Human-Centered Technologies, Università Campus Bio-Medico di Roma, Rome, Italy [3]Max Planck Institute for Intelligent Systems [4]ELLIS Institute Tübingen [5]Tübingen AI Center. Correspondence to: Omar Coser <omar.coser@unicampus.it>, Antonio Orvieto <antonio@tue.ellis.eu>.

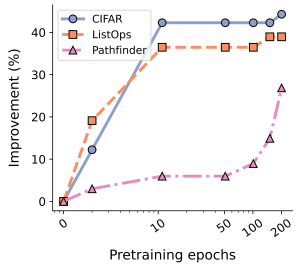
Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Table 1. Replication of the result of Amos et al. (2023). We use their codebase and similar training settings. We find good agreement with their results and hyperparameters. The numbers reported always refer to final finetuning accuracy, with the values around green parenthesis referring to the absolute performance increase with respect to the relative from-scratch result.

TRAINING MODALITY	LISTOPS	CIFAR10	PATHFINDER	RETRIVAL	TEXT
STANDARD (FROM SCRATCH)	0.402	0.499	0.67	0.776	0.653
+SELF-PRETRAINING (RESULTS FROM AMOS ET AL.)	0.59 (+0.19)	0.74 (+0.24)	0.88 (+0.21)	0.88 (+0.11)	0.89 (+0.24)
+SELF-PRETRAINING (OUR REPRODUCTION OF AMOS ET AL.)	0.56 (+0.16)	0.72 (+0.22)	0.85 (+0.18)	0.88 (+0.11)	0.89 (+0.24)

Table 2. Self-pretraining epochs ablation. We first self-pretrain with X number of epochs (full cosine annealing in that number of epochs). We then finetune for 100 epochs on each dataset. Reported are the final finetuning results after 100 epochs: we found that on ListOps, CIFAR10, and PathFinder the relative finetuning performance except for pathfinder saturates after 10 self-pretraining epochs.

PRETRAINING LEN.	LISTOPS	CIFAR10	PATHFINDER
0 EPOCHS	0.403	0.499	0.67
1 EPOCH	0.48(+0.08)	0.56(+0.06)	0.69(+0.02)
10 EPOCHS	0.55(+0.15)	0.71(+0.21)	0.71(+0.04)
50 EPOCHS	0.55(+0.15)	0.71(+0.21)	0.71(+0.04)
100 EPOCHS	0.55(+0.15)	0.71(+0.21)	0.73(+0.06)
150 EPOCHS	0.56(+0.16)	0.71(+0.21)	0.77(+0.10)
200 EPOCHS	0.56(+0.16)	0.72(+0.22)	0.85(+0.18)



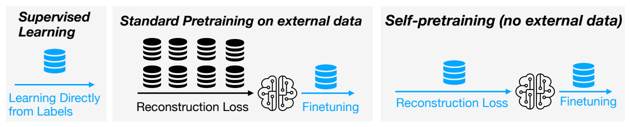

Supervised Learning
Learning Directly from Labels
Standard Pretraining on external data
Reconstruction Loss
Finetuning
Self-pretraining (no external data)
Reconstruction Loss
Finetuning

Figure 1. Illustration of different training pipelines for classification. SPT is the method by Amos et al. (2023) that we study here.

transformers benefit significantly from data-informed initialization. While this is a great practical insight, we believe it opens up intriguing questions.

Inspired by Amos et al. (2023), we here present some preliminary results and ablations on self-pretraining transformers with masked token prediction objectives, with the hope of drawing more attention to this topic – of interest both for empirical and practical research. Our objective is to *simplify* the picture by starting a discussion around the following questions:

1. Is the boost in test accuracy observed when including self-pretraining in the LRA pipeline only observable in deep models and after self-pretraining for hundreds of epochs? In other words, can we observe this effect in simpler models, perhaps with only one layer?

2. Our second question is a sanity check: Is the self-pretraining boost due to learning data-specific features, or can it also be observed when pretraining on a different dataset compared to finetuning?

3. Is poor from-scratch test performance an optimization issue (e.g., reconstruction loss allows gradients to flow better), or is it instead due to a fundamental generalization problem?

Towards answering these questions, we start by replicating the results of Amos et al. (2023) in Sec. 2. We then present ablations on pretraining budget (Sec. 3.1), model size and dataset source (Sec. 3.2). We conclude by inspecting the role of intermediate layers in Sec. 3.3, and summarize our findings and insights in Sec. 4.

We limit ourselves, in this work, to developing intuitions on top of the experimental setting by Amos et al. (2023). However, we believe it would be interesting, for future investigations, to also operate beyond the LRA.

2. Preliminaries

The LRA (Tay et al., 2020a) consists of 6 classification tasks, where if inputs are not strictly of a sequential nature (e.g. Image, the black/white version of CIFAR10) they are first flattened to produce a sequence. For a full description of these tasks, particularly comments on the increased complexity compared to baselines in the original papers cited for each dataset, please refer to Tay et al. (2020a) and the appendix.

Our first step towards a closer inspection of the results by Amos et al. (2023) is a replication of their findings, for which we use their codebase [1] and their same transformer settings (width, depth, heads, etc). We limit ourselves here to self-pretraining through a masked[2] sequence modeling objective and consider bidirectional processing with a transformer encoder (Vaswani et al., 2017) using rotary posi-

[1]https://github.com/IdoAmos/not-from-scratch

[2]Following Amos et al. (2023), 50% for images, 15% for text tasks, and 10% in ListOps.

tional embeddings (Su et al., 2021). After a 200 epochs self-pretraining stage where the model learns to reconstruct dataset-specific inputs (self-pretraining), we perform fine-tuning for 100 epochs and compare the results with those of from-scratch training on 100 epochs (no self-pretraining).

We found hyperparameters provided by (Amos et al., 2023) to be well-tuned both in the self-pretraining and in the fine-tuning stage, and their results align well with our reproduction in Table 1. This manuscript does not consider the PathX dataset since it requires significant computational efforts.

3. Ablations

We present here three set of ablations bringing insights into SPT mechanisms. We conclude in Section 4 bringing together our insights and perspectives for future work.

3.1. Self-pretraining Duration

In our experiments leading to Table 1, we noticed that pretraining is particularly computationally expensive[3]. While (Amos et al., 2023) present detailed ablations regarding dataset sizes, we are most curious of decreasing the epochs budget from 200 (baseline) down to a single epoch. Our results in Table 1 show a curious phenomenon: **only 10 epochs** of self-pretraining are needed to significantly boost performance on CIFAR10 and ListOps. Instead, for PathFinder, major gains can only be observed after 100 epochs. Additionally, we found that improvements can already be observed after a single epoch. For this experiment, our hyperparameters are kept the same as in Table 1, with the only difference in the 1-epoch setting where we reduced the learning rate warm-up settings to fit the reduced step budget.

3.2. Model Depth and Data Source

A natural explanation for the observed increase in test accuracy is the ability of self-pretraining to learn crucial patterns and **hierarchical representations**, providing a “better starting point” when learning labels (Hubel & Wiesel, 1962; Hinton & Salakhutdinov, 2006). While in Sec. 3.2, we showed how such representations may arise with just a few epochs of self-pretraining, here we inspect the efficacy of self-pretraining as we vary the **number of layers** in the model. At the same time, to test whether self-pretraining is indeed learning task-specific representations, we consider **swapping pretraining datasets**: we pretrain on Pathfinder data and fine-tune on CIFAR and vice versa.

[3]In CIFAR the wall-clock time for 1 epoch of pretraining on our hardware (a single A100 with 80GB of RAM) is 48s, for ListOps is 19.1m and for Pathfinder is 3.20min. At finetuning, the time for 1 epoch in CIFAR is 56s, for ListOps is 18.95m and for Pathfinder is 3min. Since as Amos et al. (2023) we first pretrain for 200 epochs and then finetune on 100, the total time compared to basic training from scratch for 100 epochs is tripled.

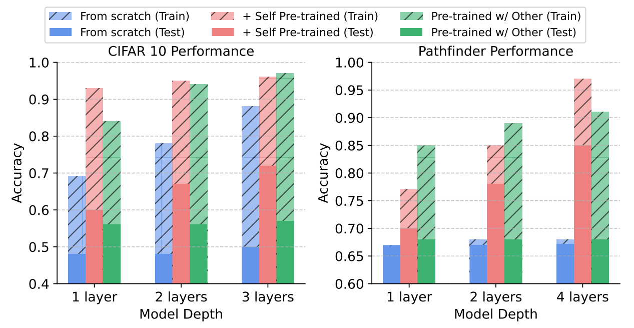


Figure 2. Settings are exactly the same as in Sec. 2 (200 pretraining epochs), comment in the main text.

For $A, B \in \{\text{CIFAR10}, \text{Pathfinder}\}$, we (1) train from scratch on data A, (2) pretrain on data A, fine-tune on data A, or (3) pretrain on data A, fine-tune on data B. In Fig. 2 we report both train and test accuracy after fine-tuning.

1. We observe **gains even when pretraining on a single layer**. While test accuracy at finetuning is only improved by 2% after self-pretraining a 1-layer model on Pathfinder, on CIFAR10 we observe a 15% increase. Even though the gap in deeper models is higher, especially for Pathfinder, we find it interesting that it can be observed at such small scales.

2. From-scratch test accuracies on both datasets are almost **independent of the number of layers**. Moreover, there is a substantial gap in training accuracy between from-scratch and SPT performance, pointing to optimization issues.

3. Some gains in test accuracy after finetuning can be **observed when switching data source**. This also points to SPT targeting, to some degree, suboptimal initialization and unlocking high training accuracy.

3.3. Freezing layers : From Scratch Sensitivity Analysis

In Sec. 3.2 we found that, SPT (regardless of data source) greatly improves training accuracy – even when models are just 1-layer deep. To further inspect this phenomenon, we start from an insight coming from vision (Touvron et al., 2021) indicating that, to improve performance, **attention layers may profit from additional supervision** beyond hard labels. To this end, we inspect the effects of freezing attention layers – i.e. of clamping operations to their value at random initialization when training from scratch. In Table 3, we observe that (with the exception of Pathfinder) **training the attention layer has little to no effect on from-scratch accuracy**: this is surprising, and indicates the *inability of attention layers to help construct useful features to solve*

Table 3. Comparison of different freezing strategies when training from scratch. Reported are test accuracy results, the setting is the same as in Sec. 2. We either clamp attention or attention and intermediate feedforward layers to random initialization. Encoder and decoder layers (linear) are always trained.

DATASET	FROM SCRATCH	+ATT. FROZEN	+ATT. & MLP FROZEN
CIFAR-10	0.50	0.56 (+0.06)	0.43 (-0.07)
PATHFINDER	0.67	0.50 (-0.17)	0.50 (-0.17)
LISTOPS	0.40	0.40 (+0.00)	0.30 (-0.10)
RETRIEVAL	0.78	0.78 (+0.00)	0.67 (-0.09)
TEXT	0.65	0.65 (+0.00)	0.66 (+0.01)

the task at hand, when learning from labels. Instead, as expected (with the exception of the Text task) freezing also the intermediate MLP layers in the transformer block always decreases performance.

4. Insights and Future Research

In Section 3, we ablated on the SPT paradigm, testing which components are necessary to observe gains. From our preliminary results, to be tested thoroughly in a full paper, we can conclude that:

(1) SPT benefits can be observed in extremely small models (1 layer) at a low number of SPT epochs (e.g., 10). While for challenging datasets like PathFinder one may require bigger models and substantial SPT effort, we find it interesting that some important gains *can be* observed even at small scale.

(2) SPT helps with optimization (training performance) in addition to generalization. This is perhaps our most crucial finding, which can be inspected in Figure 2: even at a single layer, from-scratch training fits the training data suboptimally. This hints us that some component in the architecture may be hard to optimize directly from labels.

(3) Building on top of (2), we performed a sensitivity analysis in Table 3. We found that, when training from scratch, freezing the MLP leads to a drastic decrease in performance. Instead, freezing the attention layer to random initialization can have a much milder effect–suggesting that attention layers are harder to optimize from scratch than MLPs.

Our findings point to an interesting conclusion: SPT is particularly crucial in the presence of attention layers. This also aligns well with the results of Amos et al. (2023), showing that S4 and other recurrent models do not gain much using SPT, compared to attention. Motivated by this, we present one last experiment using 1-layer models: we compare training from scratch with SPT followed by fine-tuning. As opposed to before, we now consider **initializing to SPT weights only the parameters inside the softmax layers**: the queries/keys weights. Figure 4 confirms our insights: at least for 1 layer models, the effect of SPT can be attributed to a better initialization of the attention parameters.

While the results above are non-conclusive, they point to a clear direction for theoretical research. Indeed, while it is known that deep transformers can have optimization problems (Noci et al., 2022; Wang et al., 2024), not much is known about potential issues for single layers trained from labels. We believe understanding such problems is of crucial importance for practitioners, specifically for applications where pretraining is not part of the pipeline.

5. Acknolegment

The authors acknowledge the financial support of the Hector Foundation. Omar acknowledge Proff. Paolo Soda and Proff. Loredana Zollo for the financial support.

References

Amos, I., Berant, J., and Gupta, A. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. *arXiv preprint arXiv:2310.02980*, 2023.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

Dao, T. and Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024.

El-Nouby, A., Izacard, G., Touvron, H., Laptev, I., Jegou, H., and Grave, E. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling

Table 4. For one-layer models, SPT improved test accuracy is due to better initialization of queries and keys parameters.

Dataset	From Scratch	SPT → Full Model init	SPT → Only QK init
CIFAR-10	0.48	0.60 (+0.12)	0.60 (+0.12)
Pathfinder	0.67	0.70 (+0.03)	0.70 (+0.03)

with selective state spaces. In *Conference on Language Modeling*, 2024.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022a.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Hubel, D. H. and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Krishna, K., Garg, S., Bigham, J. P., and Lipton, Z. Downstream datasets make surprisingly good pretraining corpora. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images, 2009.

Linsley, D., Kim, J., Veerabadran, V., Windolf, C., and Serre, T. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems*, 2018.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150. Association for Computational Linguistics, 2011. URL https://www.aclweb.org/anthology/P11-1015.

Nangia, N. and Bowman, S. R. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

Noci, L., Anagnostidis, S., Biggio, L., Orvieto, A., Singh, S. P., and Lucchi, A. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35: 27198–27211, 2022.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, 2023.

Peng, B., Alcaide, E., Anthony, Q. G., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M. N., Derczynski, L., Du, X., Grella, M., GV, K. K., He, X., Hou, H., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Lin, J., Mantri, K. S. I., Mom, F., Saito, A., Song, G., Tang, X., Wind, J. S., Woźniak, S., Zhang, Z., Zhou, Q., Zhu, J., and Zhu, R.-J. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics*, 2023.

Peng, B., Goldstein, D., Anthony, Q., Albalak, A., Alcaide, E., Biderman, S., Cheah, E., Du, X., Ferdinan, T., Hou, H., et al. Eagle and Finch: RWKV with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Radev, D. R., Muthukrishnan, P., Qazvinian, V., and Abu-Jbara, A. The acl anthology network corpus. *Language Resources and Evaluation*, 47:919–944, 2013.

Smith, J. T., Warrington, A., and Linderman, S. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *arXiv.org*, abs/2104.09864, 4 2021. ISSN 2331-8422. URL https://arxiv.org/abs/2104.09864.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020a.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 55:1 – 28, 2020b.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Waleffe, R., Byeon, W., Riach, D., Norick, B., Korthikanti, V., Dao, T., Gu, A., Hatamizadeh, A., Singh, S., Narayanan, D., et al. An empirical study of Mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Wang, H., Ma, S., Dong, L., Huang, S., Zhang, D., and Wei, F. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024a.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024b.

Zimerman, I. and Wolf, L. On the long range abilities of transformers. *arXiv preprint arXiv:2311.16620*, 2023.

Appendix

A. Tasks

- **ListOps:** Inspired by (Nangia & Bowman, 2018). Sequences are nested lists describing mathematical operations applied to token sets. The task requires understanding hierarchical structures. The sequence length is 2048.
- **Text:** Character-level IMDb reviews (Maas et al., 2011) for binary sentiment classification. Sequences are up to 2048 tokens.
- **Retrieval:** Character-level binary classification of document similarity scores with sequences up to 4096 tokens. The task was introduced by Radev et al. (2013).
- **Image:** Grayscale CIFAR-10 (Krizhevsky et al., 2009) images flattened into 1D sequences for 10-way classification without explicit 2D inductive bias, sequence length 1024.
- **Pathfinder, PathX:** Synthetic 2D visual tasks introduced by (Linsley et al., 2018), treated as 1D sequences for tracing capabilities (sequence lengths 1024 and 16384, respectively). They are both binary classification tasks.

B. Experimental details

Our training settings in most ablations closely follow choices by Amos et al. (2023).

- For **ListOps**, the model is trained with an embedding size of 512, input to 6 transformer layers, each with 8 attention heads. The feed-forward network has a hidden size of 1024. The tuned learning rates used for our experiments are 1×10^{-4} (finetuning) and 1×10^{-3} (pre-training), with batch sizes 64 to 128 (respectively). Weight decay is set at 0.1, and cross-entropy is used as the pretraining loss.
- In the **Text** dataset, the training strategy remains similar to ListOps, with identical feature size, depth, and attention heads. However, the learning rates are slightly adjusted to 1×10^{-4} (finetuning) and 5×10^{-4} (pre-training), with batch sizes of 64 and 32 (respectively).
- For the **Retrieval** dataset, the model adopts a smaller feature size of 128 and a reduced depth of 4 layers, with 4 attention heads and a feed-forward hidden size of 512. The learning rates are 5×10^{-4} (finetuning) and 5×10^{-3} (pre-training), with batch sizes of 16 and 32 respectively. Notably, weight decay is here set to zero, and cross-entropy loss is used.
- The Image dataset (a.k.a **CIFAR10**) employs a distinct setup, with a feature size of 64 and a shallow depth of 3 layers. There are 4 attention heads and a feed-forward hidden size of 128. Unlike the previous tasks, max pooling is used at the last layer, reflecting the spatial nature of image data. The learning rates are fixed at 1×10^{-3} (finetuning and pre-training), with batch sizes of 16 and 32(finetuning and pre-training, respectively). Weight decay is set to zero. The pretraining loss is here the L2 loss, emphasizing pixel-level reconstruction rather than classification.
- Finally, in the **Pathfinder** dataset, the model configuration closely resembles that of Retrieval, with a feature size of 128, a depth of 4 layers, 4 attention heads, and a feed-forward size of 512. The learning rates vary between 5×10^{-4} (finetuning) and 1×10^{-3} (pre-training), with batch sizes of 16 and 32 (respectively). Like ListOps and Retrieval, cross-entropy loss is used for pretraining.

During further experiments involving changing number of layers and datasets, we found these settings to be mostly stable, with only minor adjustments to the learning rate needed.

C. Further Experimental results

Table 5. Ablation on number of layers and data source, see Sec. 3.2. This data is shown in Figure 2

DATA	MODALITY	1 LAYER	2 LAYERS	3/4 LAYERS
CIFAR 10	FROM SCRATCH (TRAIN)	0.69	0.78	0.88
	FROM SCRATCH (TEST)	0.48	0.48	0.50
	+ SELF PRE-TRAINED (TRAIN)	0.93	0.95	0.96
	+ SELF PRE-TRAINED (TEST)	0.60 (+0.13)	0.67 (+0.19)	0.72 (+0.22)
	PRE-TRAINED W/ PATHFINDER (TRAIN)	0.84	0.94	0.97
	PRE-TRAINED W/ PATHFINDER (TEST)	0.56 (+0.8)	0.56 (+0.8)	0.57 (+0.7)
PATHFINDER	FROM SCRATCH (TRAIN)	0.67	0.76	0.83
	FROM SCRATCH (TEST)	0.669	0.670	0.672
	SELF PRE-TRAINED (TRAIN)	0.77	0.85	0.97
	SELF PRE-TRAINED (TEST)	0.70 (+0.03)	0.78 (+0.10)	0.85 (+0.18)
	PRE-TRAINED W/ CIFAR 10 (TRAIN)	0.85	0.89	0.91
	PRE-TRAINED W/ CIFAR 10 (TEST)	0.68 (+0.01)	0.68 (+0.01)	0.68 (+0.01)