Towards Understanding Self-Pretraining for Sequence Classification with Transformers

Omar Coser* **Loredana Zollo†** **Paolo Soda‡** **Antonio Orvieto§**

Abstract

Amos et al. (2024) showed that the accuracy of Transformer models in sequence classification can be significantly improved by first pretraining with a masked token prediction objective *without external data or augmentation*, a procedure referred to as self-pretraining (SPT). While the primary objective of Amos et al. (2024) was to showcase that Transformers can achieve strong performance on the Long-Range Arena (LRA), their pipeline raises more fundamental questions: How does SPT drive optimization to better solutions? Why can standard supervised training fail in Transformers? To better understand this, we replicate and systematically ablate the findings of Amos et al. (2024). Our ablations suggest that a central bottleneck in the studied settings is not depth or generalization alone, but the ability of label supervision to learn useful query-key Attention patterns from random initialization. With a minimal setup, we identify learning proximity interactions – turning absolute positional encodings into proximity-biased Attention scores – as a key source of the improvements brought by SPT. Finally, in a simplified theoretical setup, we show that label supervision can be locally blind to certain Attention-score directions that are instead detectable through masked reconstruction.

1 Introduction

The Long-range Arena [Tay et al., 2020] has played an important role in the development of new efficient token-mixing strategies over the last few years. A prime example is S4 [Gu et al., 2022b], the first state-space models (SSMs), which drew attention after surpassing Transformers [Vaswani et al., 2017] on the LRA, achieving a 20% average increase in accuracy. S4 and its later variants, many of which were developed and evaluated using the LRA benchmark [Smith et al., 2023, Gu et al., 2022a, Poli et al., 2023, Orvieto et al., 2023], laid the groundwork for modern architectures such as Mamba [Gu and Dao, 2024, Dao and Gu, 2024], GLA [Yang et al., 2024a], RWKV [Peng et al., 2024, 2025] and (Gated) DeltaNet [Yang et al., 2024b,a] among others. While the latest developments are mainly inspired by language modeling performance [Waleffe et al., 2024, Poli et al., 2024], researchers have maintained interest in explaining and studying the gap between Transformers and SSMs on LRA, see e.g. Zimerman and Wolf [2023], Alonso et al. [2025], Qin et al. [2024].

In their seminal contribution, awarded the *Outstanding Paper Award at ICLR 2024*, Amos et al. [2024] 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

*Unit of Artificial Intelligence & Computer Systems, Università Campus Bio-Medico di Roma, Via Álvaro del Portillo, 21, Rome, 00128, Italy. Unit of Advanced Robotics and Human-Centered Technologies, Università Campus Bio-Medico di Roma, Via Álvaro del Portillo, 21, Rome, 00128, Italy. Corresponding author: omarcoser10@gmail.com.

†Unit of Advanced Robotics and Human-Centered Technologies, Università Campus Bio-Medico di Roma, Via Álvaro del Portillo, 21, Rome, 00128, Italy.

‡Unit of Artificial Intelligence & Computer Systems, Università Campus Bio-Medico di Roma, Via Álvaro del Portillo, 21, Rome, 00128, Italy. Department of Diagnostics and Intervention, Radiation Physics, Biomedical Engineering, Umeå University, Universitetstorget, 4, Umeå, 40196, Sweden.

§Max Planck Institute for Intelligent Systems. ELLIS Institute Tübingen. Tübingen AI Center.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

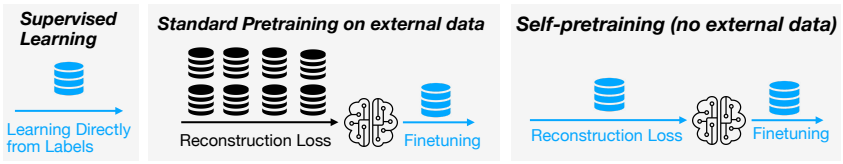


Figure 1: Training pipelines for classification. SPT is the method we study here.

pretraining → finetuning setup (see Tb. 1). At first glance, this finding may not seem too surprising – yet, **pretraining is performed here on the same data** (self-pretraining, denoted as **SPT**), that is: for a fixed task, the Transformer is first asked to learn to predict masked tokens (or the next token) in the sequence, and only later is trained on task labels. This procedure differs substantially from the usual pretraining paradigm (a large, heterogeneous pretraining corpus and a small, specialized fine-tuning dataset, e.g., Brown et al. [2020]).

However, the idea of SPT (Fig. 1) is not attributable to Amos et al. [2024]: it can already be found in El-Nouby et al. [2021], who showed the efficacy of a self-pretraining pipeline in vision. Self-pretraining as a new paradigm then appeared in Peng et al. [2023], showing that SPT models achieve competitive performance on downstream speaker verification tasks with only one-third of the data compared to standard pretraining. Concurrently, Zhou et al. [2023] demonstrated that self-pretraining improves performance across diverse medical image tasks. Further, Krishna et al. [2023] demonstrated that pretraining language models directly on downstream datasets could match performance after 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 and its heterogeneity*.

Table 1: **Replication of the results** of Amos et al. [2024]. We use their codebase and similar training settings, see § 2. We find good agreement with their results and hyperparameters. The numbers reported always refer to final peak held-out validation accuracy, with the values in green parentheses referring to the absolute performance increase with respect to the corresponding from-scratch result.

Training Modality	ListOps	CIFAR10	PathFinder	Retrieval	Text
Standard (from scratch)	0.402	0.499	0.67	0.776	0.653
+SPT (Amos et al.)	0.59 (+0.19)	0.74 (+0.24)	0.88 (+0.21)	0.88 (+0.11)	0.89 (+0.24)
+SPT (Our reproduction)	0.56 (+0.16)	0.72 (+0.22)	0.85 (+0.18)	0.88 (+0.11)	0.89 (+0.24)

Compared to the applied research cited above, the objective of Amos et al. [2024] was not to show how to reach state-of-the-art performance or to eliminate the need for pretraining on a massive external dataset. Instead, they show that on challenging tasks such as the LRA, **Transformers, in contrast to other models such as SSMs, *critically* benefit from data-informed initialization in supervised learning**. This is a valuable practical insight; it **hints at a potential difficulty in learning Attention masks from labels alone**, an interesting phenomenon that warrants further investigation.

To shed light on the issue, we first pose the following questions that isolate depth, data source, and optimization effects. We mostly explore these on the LRA, as it is challenging for Attention-based models [Tay et al., 2020, Zimerman and Wolf, 2023, Amos et al., 2024].

1. *Role of depth:* Is the boost in validation accuracy observed when including SPT only observable in deep models and after a self-pretraining budget of hundreds of epochs? Can we observe and try to understand this effect in shallow models, perhaps with only one layer?
2. *Role of data:* Is the self-pretraining boost due to learning data-specific features, or can it also be observed when pretraining e.g. on another (small-scale) dataset in the same benchmark?
3. *Role of optimization:* Is poor from-scratch validation performance an optimization issue (e.g., reconstruction loss allows gradients to flow better), or is it instead the case that both from-scratch training and self-pretraining fit the data, yet SPT leads to better generalization?

Towards answering these questions, we make the following contributions:

- We replicate the results of Amos et al. [2024] in Tb. 1, and show additional evidence for SPT beyond the LRA in § B.1. We then present ablations regarding pretraining duration (§ 3.1), model depth and data source (§ 3.2), weight freezing (§ 3.3) and mixed initializations (§ 3.4).
- Throughout § 3 we summarize our main findings in "Takeaway boxes". Our results show that SPT gains persist across several LRA settings (various model depths, different data sources and

pretraining durations). By re-initializing subsets of model weights to a uniform distribution after self-pretraining and before finetuning, our ablations identify Attention parameters, and especially W_Q, W_K in one-layer models, as major carriers of the SPT gain.

- We further investigate the Attention patterns unlocked by SPT. To do that, we construct in § 4 a minimal 1-layer synthetic example. A careful analysis of this setup reveals that self-pretraining can learn Attention weights W_Q, W_K that convert additive sinusoidal positional encodings into a relative, proximity-biased Attention pattern that finetuning can build on.
- Why can learning specific Attention patterns fail with pure label supervision? In § 5 we provide a theoretical abstraction and analysis in a simplified yet insightful setup.

We believe our investigation can spark further theoretical discussions of the statistical and optimization properties of this promising new paradigm.

2 Preliminaries

Model. Let B denote the batch size, L the sequence length and D the embedding dimension. Given embedded inputs $X \in \mathbb{R}^{B\times L\times D}$, Amos et al. [2024] inject absolute positional information once as $Z^{(0)} = X + P$. The backbone is a pre-norm [Xiong et al., 2020] residual stack of T blocks, each consisting of an Attention layer followed by a position-wise MLP. For each block, $Z' = Z + \mathrm{Drop}(\mathrm{MHA}(\mathrm{LN}(Z)))$ and $Z^+ = Z' + \mathrm{Drop}(\mathrm{MLP}(\mathrm{LN}(Z')))$. Attention is multi-head: for H heads, $d = D/H$, $Q_h = Z(W_Q^h)^\top$, $K_h = Z(W_K^h)^\top$, $V_h = Z(W_V^h)^\top$, the operation is

$$\mathrm{MHA}(Z) = \mathrm{Concat}_h\left(\mathrm{softmax}\Big(\tfrac{Q_h K_h^\top}{\sqrt{d}}\Big)V_h\right)W_O.$$

For classification, the sequence $Z^{(T)}$ is mean-pooled, and the output vector is used for classification. In self-pretraining, the architecture is exactly the same, yet a fixed fraction $\mathcal{M}$ of the input sequence is masked and the model is trained to reconstruct its original continuous values. $Z^{(T)}$ is projected token-wise and evaluated using $\mathcal{L}_{\mathrm{SPT}} = \sum_{b=1}^{B}\sum_{t\in\mathcal{M}} \ell\Big(W_{\mathrm{spt}} Z^{(T)}_{b,t,:},\ X_{b,t,:}\Big)$.

Training setup. The LRA 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, please refer to our Appendices A.1 and A.2. We do not consider the PathX dataset, where the full SPT+finetuning pipeline is roughly $16\times$ more expensive than PathFinder. Our LRA conclusions thus apply to the other five tasks, not the full six-task benchmark. Our first step towards a closer inspection of the results by Amos et al. [2024] is a replication of their findings, for which we use their codebase and the same Transformer settings (width, depth, heads, etc). We follow Amos et al. [2024] and consider self-pretraining through a masked[1] sequence modeling objective. After a 200 epochs SPT stage where the model learns to reconstruct dataset-specific inputs (self-pretraining), we perform finetuning for 100 epochs and compare the results with those of from-scratch training for 100 epochs (no SPT). We use AdamW [Loshchilov et al., 2017] and found the hyperparameters provided by Amos et al. [2024] to be well-tuned both in the self-pretraining and finetuning stages, and their results align well with our reproduction in Tb. 1.

Following prior LRA work, we use the public held-out split as the validation/evaluation split. Following Amos et al. [2024], scalar LRA results report the *peak held-out accuracy* attained within the fixed finetuning budget. For the CIFAR 10 and PathFinder ablations that support our Attention-specific claims, namely the freezing, hybrid-initialization, and one-layer QK experiments, we report mean $\pm$ standard deviation over 5 random seeds. For the ListOps, Retrieval, and Text entries of the freezing experiment (Tb. 2), we report mean $\pm$ standard deviation over 3 random seeds. Results without a $\pm$ value, including the larger LRA sweeps, are single-seed runs due to computational cost and low observed variation in the maximum accuracy over iterations on CIFAR 10 and PathFinder.

To motivate our study beyond the LRA, we provide evidence of the SPT gap on *additional datasets* in §B.1, covering human activity and physiological signal classification [Dempster et al., 2019].

[1] Following Amos et al. [2024], 50% for images, 15% for text tasks, and 10% in ListOps.

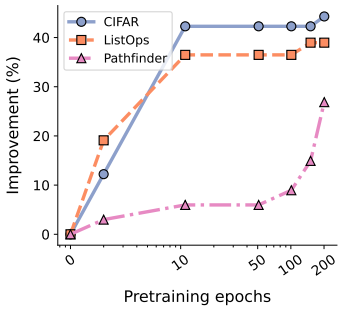


Figure 2: **SPT duration** ablation. We vary self-pretraining epochs before 100 epochs of finetuning. CIFAR10 and ListOps benefit after only a few epochs, PathFinder needs longer pretraining.

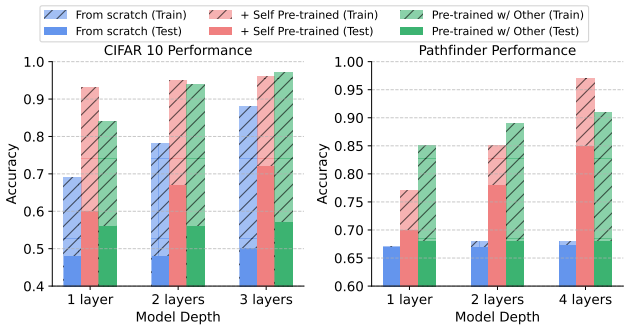


Figure 3: **Depth and dataset-swap** ablation. We compare training from scratch, same-dataset SPT, and cross-dataset pretraining before finetuning. SPT can improve both train and test accuracy even for 1-layer models and with swapped pretraining data, pointing to an optimization benefit beyond task-specific representation learning.

3 Ablations

We present here four sets of ablations on the LRA, bringing insights into SPT mechanisms: varying pretraining duration (§ 3.1), changing model depth and data source (§ 3.2), freezing layers when training from scratch (§ 3.3), and using only a subset of self-pretrained weights (§ 3.4).

3.1 Self-pretraining Duration

In our experiments leading to Tb. 1, we noticed that pretraining can be computationally expensive[2]. Our results in Fig. 2 show that *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. Improvements can already be observed after a single epoch. Our hyperparameters are kept the same as in Tb. 1, except in the 1-epoch setting where we reduced the learning rate.

Takeaway 1: SPT can yield substantial improvements even after a single epoch of pretraining.

3.2 Evidence for an optimization bottleneck (Depth + Dataset Swap)

A natural explanation for the observed increase in validation accuracy could be the ability of self-pretraining to learn crucial patterns and **hierarchical representations** of the data, providing a *better data-distribution-informed initialization* when learning labels [Hubel and Wiesel, 1962, Hinton and Salakhutdinov, 2006]. While in § 3.1, 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 — testing whether SPT success is contingent on the presence of a hierarchy in representations. At the same time, to test whether self-pretraining is indeed learning task-specific representations, we consider **swapping pretraining datasets**.
For each target dataset A, we compare: training from scratch on A, SPT on A followed by finetuning on A, and pretraining on another dataset $B \neq A$ followed by finetuning on A. Results are in Fig. 3.

1. We observe **gains even when pretraining on a single layer**. While validation accuracy after fine-tuning improves by only $\sim 2\%$ after self-pretraining a 1-layer model on PathFinder, on CIFAR10 we observe a $\sim 15\%$ increase. Even though the gap in deeper models is larger, especially for PathFinder, we find it interesting that it can be observed at such small scales.
2. On both datasets, **from-scratch validation accuracies 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.

[2]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. During 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. [2024] 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.

3. Some gains in validation accuracy after finetuning can be **observed when switching data source**. This suggests that SPT may improve trainability by moving the model away from a suboptimal initialization, rather than only by learning task-specific representations.

Compute-matched and tuned from-scratch baseline. To rule out that the gap is due to the smaller total budget or to a learning rate tuned for SPT, we train from scratch for 400 epochs (more than the total SPT+finetuning budget), sweeping the learning rate over four orders of magnitude under both a constant schedule with warmup and cosine decay with warmup (App. C.1). The best configuration reaches only 0.501 on CIFAR10 and 0.685 on PathFinder, far below SPT (0.724 and 0.85), and validation accuracy typically peaks within ~ 50 epochs before degrading. The full SPT pipeline is also cheaper in wall-clock time than this extended baseline (App. C.2).

Takeaway 2: The SPT boost is not necessarily related to generalization: SPT can greatly improve *training* accuracy after finetuning – even when models are just 1-layer deep. This can also hold when pretraining on a different data distribution from another LRA task. This hints at optimization problems when training Attention-based sequence models directly from labels.

3.3 Freezing layers: from-scratch sensitivity analysis

In § 3.2, we saw that short pretraining, either on task data or on data with a different distribution, can improve training accuracy (and sometimes, validation accuracy) when learning from labels. Amos et al. [2024] observed that models with different sequence mixers, such as SSMs, can be less sensitive to SPT. This suggests that the trainability issue may be tied to softmax Attention itself. To this end, we consider a drastic experiment: training from scratch after freezing all Attention weights in the architecture. If attention is not effectively learned from labels, then freezing Attention at random initialization should have little effect on performance. This finding would also be consistent with insights from early-stage vision Transformer training [Touvron et al., 2021], indicating that Transformers are data-hungry and may benefit from additional supervision beyond hard labels.

In Tb. 2, training the Attention layers has little effect when training from scratch, except on Pathfinder. In some cases, such as CIFAR-10, freezing Attention even improves performance. By contrast, also freezing the intermediate MLP layers usually decreases performance, as expected.Multi-seed runs on ListOps, Retrieval, and Text confirm this picture: freezing Attention changes mean accuracy by at most 0.01, while additionally freezing the MLP degrades ListOps (-0.12) and Retrieval (-0.11) and leaves Text unaffected. Variance is larger when both Attention and MLP are frozen, reflecting less stable training when most parameters are clamped, but the degradation is consistent across seeds.

Table 2: Comparison of different **freezing configurations** when training deep models **from scratch**. Reported are val. accuracy results, in the setting described in § 2. We either clamp only Attention weights or Attention weights and intermediate feedforward layers to random initialization, and learn other parameters from labels. Encoder, normalization weights, and decoder layers are always trained. CIFAR10 and PathFinder entries report mean $\pm$ standard deviation across 5 seeds; ListOps, Retrieval, and Text entries report mean $\pm$ standard deviation across 3 seeds.

DATASET	FROM SCRATCH	WITH ATT. FROZEN	WITH ATT. & MLP FROZEN
CIFAR10	0.503 ± 0.002	0.535 ± 0.004 (+0.032)	0.468 ± 0.008 (-0.035)
PATHFINDER	0.672 ± 0.003	0.50 ± 0.00 (-0.172)	0.50 ± 0.00 (-0.172)
LISTOPS	0.41 ± 0.003	0.42 ± 0.005 (+0.01)	0.29 ± 0.009 (-0.12)
RETRIEVAL	0.77 ± 0.005	0.78 ± 0.004(+0.01)	0.66 ± 0.011 (-0.11)
TEXT	0.65 ± 0.004	0.65 ± 0.006 (+0.00)	0.66 ± 0.002 (+0.01)

Takeaway 3: With the exception of PathFinder, from-scratch performance is unaffected by freezing Attention weights to random. This indicates that supervised training makes limited effective use of the Attention parameters in these settings and struggles to combine sequential information.

3.4 Using SPT initialization only on a subset of weights

§ 3.3 suggests that the gains from self-pretraining are not uniformly distributed across model components, with the Attention mechanism emerging as particularly important. Motivated by this observa-

tion, we study hybrid initializations in which, for each layer, only selected subsets of parameters are initialized from SPT weights, while all remaining parameters are randomly initialized.

The results in Tb. 3 show a clear separation between architectural components. **Initializing only MLP-related parameters yields little to no improvement over random initialization** (see *MLP+Norm+Encoder*), indicating that self-pretraining provides limited transferable signal to the MLP branch. **In contrast, initializing the Attention parameters has a pronounced effect** (see *Attention+Norm+Encoder*). In particular, using self-pretrained values solely for the Attention weights W_K, W_Q, W_V, and W_O can already lead to substantial accuracy gains in CIFAR10. While performance close to full self-pretraining is achieved only when Normalization and embedding parameters (both close in proximity to Attention) are also initialized from self-pretrained weights, these results demonstrate that the Attention block is the primary carrier of the benefits of SPT. A dedicated hyperparameter search (learning rate, warmup, weight decay) for the Att+Norm+Enc setting on PathFinder (App. C.3) reaches at most 0.781, still well below full SPT initialization (0.85), so the remaining gap is not an artifact of tuning.

To interpret the remaining gap, we note that Tb. 3 uses multilayer architectures. In this setting, pretrained Attention weights may not transfer when the surrounding weights are reset. The Attention map in layer ℓ is determined not only by W_Q and W_K, but also by representations entering that layer. Re-initializing embeddings, normalizations, or earlier-layer weights changes this representation, including its scale and distribution. Thus, pretrained W_Q, W_K may no longer produce the Attention patterns they learned during SPT. This mismatch can compound across layers, since a distorted Attention pattern in one layer alters the inputs of the next layer. Therefore, Attention-only initialization can recover part of the SPT benefit, but should not be expected to match full SPT in deeper models.

Motivated by this explanation and by § 3.2, we run a final experiment with **1-layer models**, where such cross-layer mismatch cannot accumulate. We initialize only the parameters that determine the softmax Attention scores, namely W_Q and W_K, from SPT weights. Tb. 4 confirms our interpretation: in 1-layer CIFAR10/PathFinder models, gains can be attributed to a better initialization of W_Q, W_K.

Table 3: Please refer to § 2 for our setup. We report mean and standard deviation across 5 seeds. Models are initialized in a **hybrid fashion**: we consider, for all layers, initializing to SPT values **only** (1) the Attention weights (W_K, W_Q, W_V, W_O) (2) the Attention weights, their LayerNorm parameters, and the initial embedding layer, ..., (5) Attention and MLP weights, and so on. All weights which are not initialized from SPT are initially random. All weights are then optimized with the same setting as Tb. 1. "No Init" and "Full Init" here correspond to the from scratch and SPT variant in Tb. 1, respectively. We show classification accuracy, and notice that substantial gains from random initialization can be obtained by initializing to self-pretrained weights only the parameters in the Attention branch, while specialized initialization of the MLP branch has little to no effect.

Baselines and attention-related initialization							
Dataset	**Stat.**	**None**	**Full**	**Att**	**Att+Norm**	**Att+Enc**	**Att+Norm+Enc**
CIFAR10	mean	0.503	0.721	0.532	0.515	0.578	0.714
	std.	±0.002	±0.012	±0.004	±0.024	±0.009	±0.002
PathFinder	mean	0.672	0.850	0.686	0.677	0.709	0.781
	std.	±0.003	±0.004	±0.005	±0.007	±0.009	±0.008

MLP-related and mixed initialization						
Dataset	**Stat.**	**Att+MLP**	**Att+MLP+Enc**	**Att+MLP+Norm**	**MLP+Norm**	**MLP+Norm+Enc**
CIFAR10	mean	0.540	0.503	0.504	0.487	0.495
	std.	±0.008	±0.006	±0.011	±0.004	±0.009
PathFinder	mean	0.685	0.703	0.661	0.661	0.669
	std.	±0.008	±0.005	±0.010	±0.007	±0.005

Takeaway 4: Downstream benefits can be observed even when self-pretraining is applied to only a subset of parameters. In line with Takeaway 3, we find that retaining self-pretrained initialization in the Attention layers is particularly impactful, in contrast to MLP layers. Once depth-related confounding effects are removed, SPT initialization of W_Q and W_K alone is sufficient to explain the performance gains attributed to self-pretraining.

Table 4: For one-layer models, SPT improves val. accuracy due to better initialization of W_Q, W_K. We report mean and standard deviation across 5 seeds. A full ablation is provided in Tb. 9.

DATASET	FROM SCRATCH	SPT → FULL MODEL INIT	SPT → ONLY QK INIT	SPT → ALL BUT NOT QK INIT
CIFAR10	0.486 ± 0.003	0.603 ± 0.005 (+0.12)	0.607 ± 0.012 (+0.12)	0.502 ± 0.008 (+0.016)
PATHFINDER	0.672 ± 0.002	0.705 ± 0.004(+0.03)	0.704 ± 0.003 (+0.03)	0.679 ± 0.007 (+0.008)

How much do weights move? To complement the mixed-initialization results, we compare Frobenius displacements between random initialization (R), self-pretraining (SPT), supervised training from scratch (SC), and finetuning (FT) (§ B.5). Across CIFAR10 and PathFinder, displacements $R \to SC$ are generally much smaller than $R \to SPT$ or $R \to FT$: SC stays closer to initialization. Fig. 9 further shows that from-scratch norm changes are dominated by MLP blocks, while Attention projections move less, especially relative to MLPs on PathFinder. This supports the view that SPT provides a useful Attention structure not recovered by supervised training alone.

4 A synthetic task for understanding self-pretraining on Attention weights

In § 3, we ablated the Self-Pretraining (SPT) paradigm on the LRA. We showed that SPT gains are tied to successful optimization (Takeaway 2), can be observed even in 1-layer, and are especially strong when SPT initializes the Attention weights (Takeaway 4, Tb. 3 and Tb. 4). When randomly initialized and trained with labels, these weights struggle to establish token relations (Takeaway 3, Tb. 2). Here, we move beyond LRA and introduce a minimal synthetic task that exposes the benefits of SPT in a controlled setting. This setup enables smaller models and finer ablations, and unlocks insights into how SPT acts on Attention: we find that SPT reshapes how the model uses additive positional encodings and effectively converts them into a flexible proximity bias.

Task and Data. We generate a synthetic binary classification dataset of length-100 sequences whose statistics depend on a binary label. For each label, two anchor points define the region where trajectories concentrate: each datapoint is latently an ellipsoidal trajectory moving back and forth between these anchors. To make the task non-trivial, trajectories vary in speed via random-phase time reparametrization, orbital amplitude, and orientation, and are further corrupted by Gaussian noise and outliers. § D describes the task and provides example plots. The dataset contains 100 balanced training sequences with 15% random label flips, 200 balanced validation sequences, 400 balanced test sequences, and 12000 unlabeled sequences for self-supervised pretraining.

Model. We use a one-layer Attention module to predict the label. The module is much simpler than the one used in § 3: two-dimensional trajectories are linearly embedded into 32 dimensions and processed by single-head softmax Attention with weights W_Q, W_K, W_V, W_O. There is no bias, no MLP after the Attention block, and no LayerNorm. Crucially, we make use of absolute positional embeddings, added to the encoded input before the Attention block exactly as in § 3.

4.1 Results and discussion

In Fig. 4, we consider 20 epochs of training from labels, preceded by a potential SPT (10 epochs) using a masked objective (as in § 3). For each learning rate, we gather performance over 10 seeds, and show the average along with the max-min range. **We observe consistent benefits from self-pretraining in this controlled setting**, with a boost of $\sim$ 10% in both training and test accuracies at the optimal learning rates. We also provide plots for the training loss and a different epoch budget in § D.

Attention Matrix. We proceed to inspect how the Attention matrix evolves from random initialization during pretraining. To do this, we inspect runs associated with the learning rate maximizing the gap in training loss, and then check if similar findings can also extend to the LRA in § 4.2. In Fig. 5, we see a clear pattern emerging: when feeding a single input into the self-pretrained Attention

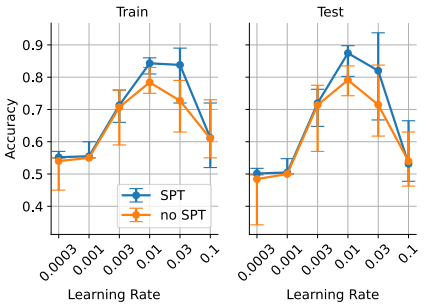


Figure 4: Performance on the toy task (1-layer model) described in § 4. Train and test accuracies are averaged over 10 random seeds (max-min interval is shown) for different learning rates. SPT consistently outperforms the no-SPT baseline, achieving higher peak accuracy at intermediate learning rates.

mechanism, a **diagonally dominated structure emerges** resembling ALiBi [Press et al., 2021] and thus drawing a direct link to the findings of Zimerman and Wolf [2023] on the LRA. Compared to random initialization of W_Q, W_K, which produces a near-uniform Attention pattern, as the softmax collapses to a flat distribution Noci et al. [2022], SPT establishes a clear proximity bias: each token attends predominantly to its neighbors.

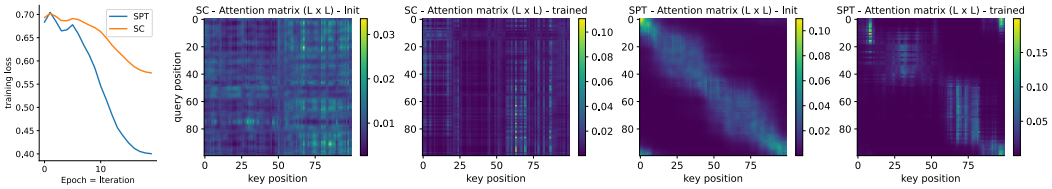


Figure 5: Toy Attention evolution. Left: training loss over iterations for self-pretraining (SPT) and From-Scratch training (SC). Right: Attention matrices ($L \times L$) at initialization and after training. SPT rapidly develops structure during pretraining, exhibiting a proximity bias. Compared to random initialization, this structure can develop into a richer sequence mixer after finetuning ("trained").

Crucial role of Positional Encodings. Motivated by the structure pretraining imposes on the Attention matrix, we inspect how this pattern emerges and its relation to additive positional embeddings. To isolate the role of positional information, we set the input content to zero and feed only positional encodings through the Attention block (Fig. 6). The same diagonal pattern as Fig. 5 emerges, indicating a data-independent mechanism. **SPT Attention weights are in this setting effectively "undoing" sinusoidal positional encodings**, building up a structure which instead allows for direct comparisons of nearby tokens already during the first finetuning steps, accelerating convergence.

Although the marginal distributions of W_Q and W_K remain close to random initialization in Fig. 6, their product $W_Q^\top W_K$ develops a structured interaction with the positional embeddings. As shown in Fig. 12, the distributions of W_Q, W_K do not exhibit any particular shift from random initialization – yet their product does, and is able to act on positional embeddings.

4.2 Bringing back insights to the LRA

The toy task in the previous subsection suggests self-pretraining effectively builds a diagonally-dominated Attention structure (Fig. 5) through a direct action on the positional encodings controlled by the product $W_Q^\top W_K$ (Fig. 6). We next ask whether the same pattern appears on the LRA. To do this, we return to the simplest setup in § 3, i.e., the one of Tb. 4.

Attention Structure. Fig. 7 shows the query-key product and the resulting softmax Attention matrix obtained after feeding a CIFAR10 or a PathFinder example to the self-pretrained network. The resulting pattern is strikingly similar to Fig. 5: pretrained W_Q, W_K establish a diagonal-dominated Attention.

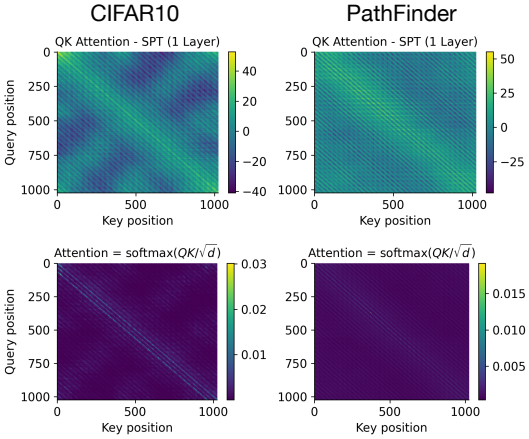


Figure 7: Visualization of raw $QK^\top$ (top) and softmax-normalized attention weights (bottom) for a single-layer self-pretrained Transformer. $QK^\top$ matrices show clear diagonal structure, with noisier patterns for CIFAR10 and more coherent structure for PathFinder. Softmax normalization yields sparse, predominantly diagonal attention. Weights are taken after SPT (before finetuning) for the 1-layer setting described in Tb. 4.

Different PE. At this point, a natural question emerges: rather than relying on SPT to bend W_Q, W_K into producing a proximity bias, why not impose such a bias directly at initialization? Instead of using absolute positional encodings, one can inject positional structure directly into Attention: writing S as the softmax argument, ALiBi [Press et al., 2021] sets $S_{ij} \mapsto S_{ij} - m|i-j|$, hard-coding a preference for nearby tokens. FIRE [Li et al., 2023] learns the shape of the bias $S_{ij} \mapsto S_{ij} + \psi_\theta(|i-j|)$. RoPE [Su et al., 2024] instead modifies the dot product itself by rotating queries and keys, making the score depend on relative position $j - i$ without adding an explicit bias. Tb. 5 shows that ALiBi can improve

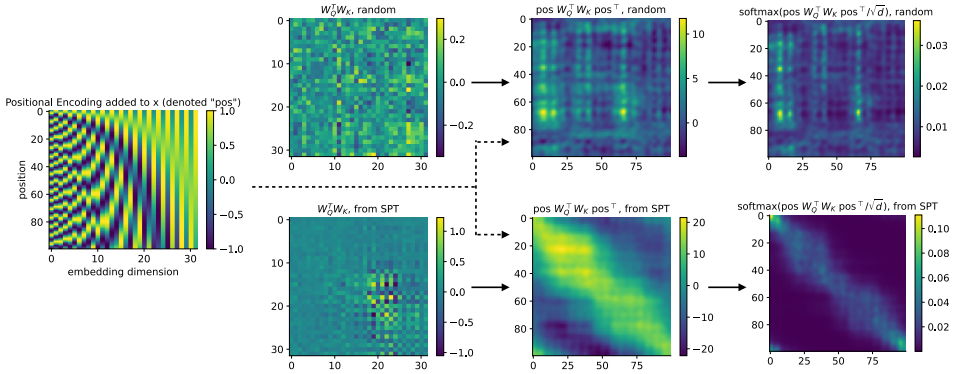


Figure 6: Toy position undoing. Visualization of Attention components with random initialization (top) and after SPT training (bottom). While random weights fail to recover positional structure, SPT learns weights that effectively undo positional encoding, producing coherent, position-aligned Attention after softmax. Here we set the input content $X = 0$ and feed only positional embeddings pos through Q/K to isolate the effect of positional information.

Table 5: Same setup as Tb. 1, ablating the effect of positional encoding variants on self-pretraining accuracy. We report from-scratch (FS) and self-pretraining (SPT) accuracy on a single run.

DATASET	STRATEGY	PE	ALIBI	FIRE	ROPE	PE+ALIBI	PE+FIRE	ROPE+ALIBI
CIFAR10	FS	0.501	0.564	0.594	0.487	0.588	0.596	0.559
	SPT	0.722	0.575	0.746	0.741	0.754	0.760	0.759
PATHFINDER	FS	0.671	0.500	0.500	0.679	0.678	0.674	0.661
	SPT	0.8506	0.500	0.500	0.71	0.9007	0.714	0.782

from-scratch trainability on CIFAR10, but it does not unlock PathFinder performance, consistent with Zimerman and Wolf [2023]. The effect is not specific to ALiBi, suggesting that the bottleneck is not access to the right positional information, but learning the right Attention bias. CIFAR10 can benefit from simple locality priors since nearby visual statistics are preserved after flattening. PathFinder instead requires tracing global connectivity; deciding whether the points are connected in 2D can require much longer-range sequence interactions. A fixed or poorly learned bias can prevent the model from forming the necessary global connections. SPT is the only strategy we found to reliably unlock this regime. We return to why this is the case in the next section.

5 Why label supervision can fail to learn Attention maps

In § 3 we presented a sequence of ablations showing that SPT is particularly helpful for learning attention weights. In § 4, we showed that SPT induces learning of a proximity bias through matrices W_Q and W_K, moving from random uniform attention scores to a diagonally-dominated token mixing. The experiments show what SPT-learned Attention maps look like, but not why label supervision may fail to learn them. We now give a simplified theoretical explanation.

Consider Attention magnitudes $a_{ij} = \exp(s_{ij}) / \sum_{k=1}^{L} \exp(s_{ik})$, parametrized by scores s_{ij}. We assume s_{ij} to be independent of the input, an abstraction motivated by Fig. 6 and 7. We consider learning a target pattern $s^*_{ij} = \Delta_{ij}$. To simplify the analysis, we study one-dimensional perturbations: $s_{ij}(\alpha) = \alpha\Delta_{ij}$. The target score pattern corresponds to $\alpha = 1$, while $\alpha = 0$ models uniform Attention initialization.

Proposition 1. *Let $X_1, \dots, X_L \in \mathbb{R}^d$ be centered random tokens with finite second moments, and define the autocorrelation matrix $C \in \mathbb{R}^{L\times L}$ as $C_{ij} := \mathbb{E}[X_i^\top X_j]$. For a score pattern $\Delta \in \mathbb{R}^{L\times L}$, consider training a single parameter α controlling the Attention score $s_{ij}(\alpha)$. Outputs are computed as $o_i(\alpha) = \sum_j a_{ij}(\alpha) X_j$. For a label Y, consider the mean-pooled (from scratch, representations are aggregated) loss $\mathcal{L}_{\text{sup}}(\alpha) = \mathbb{E}[\ell(\frac{1}{L}\sum_i o_i(\alpha), Y)]$ and the SPT loss $\mathcal{L}_{\text{SPT}}(\alpha) = \frac{1}{2L}\sum_i \mathbb{E}[\|X_i - o_i(\alpha)\|^2]$. Let $\mathbf{1} \in \mathbb{R}^L$ be the all-ones vector and define $\mathcal{B} := \{\Gamma \in \mathbb{R}^{L\times L} : \Gamma^\top \mathbf{1} = 0\}$. Define $\langle A, B\rangle_F = \sum_{i,j} A_{ij}B_{ij}$ and $\bar{\Delta} = \Delta - \frac{1}{L}\Delta\mathbf{1}\mathbf{1}^\top$. For every $\Delta \in \mathcal{B}$,*

$$\mathcal{L}'_{\text{sup}}(0) = 0, \qquad \mathcal{L}'_{\text{SPT}}(0) = -\tfrac{1}{L^2}\langle \bar{\Delta}, C\rangle_F.$$

An analogous statement holds for the case where token-to-same-token interactions are excluded.

Proposition 1 isolates a local difference between label supervision and a squared reconstruction surrogate. At uniform Attention, mean pooling cancels the first-order effect of certain score perturbations, whereas reconstruction detects them when $\langle \bar{\Delta}, C \rangle_F \neq 0$. The supervised cancellation holds for every input and does not require white-noise tokens. The proposition uses unmasked reconstruction; the diagonal-excluded variant is given in the appendix. Fig. 14 illustrates the result numerically.

Extensions and scope. Appendix E.1 extends the cancellation argument to input-dependent scores and multiple query-key parameters, and gives a corresponding condition for general token mixers. We also examine a simplified CLS readout with identical incoming token representations, and contrast this with finite causal convolution kernels, which remain distinguishable after mean pooling when all inputs are considered.

6 Conclusions

We studied self-pretraining (SPT) for Transformer sequence classification with the goal of understanding why masked reconstruction can improve supervised training. Across LRA ablations and a controlled synthetic task, our evidence points to Attention-score learning as a central bottleneck: label training often fails to organize useful query-key interactions. In our toy task and in one-layer CIFAR10/PathFinder models, we identified that SPT provides a flexible, proximity-biased Attention structure that fine-tuning can build on. Our theory provides one reason this can happen, showing that mean-pooled supervision can be locally blind to certain directions in the Attention scores that masked reconstruction can detect.

Limitations. The most detailed multi-seed component ablations focus on CIFAR10 and PathFinder; PathX is excluded for compute reasons, and several large LRA runs are single-seed. The toy task and theory deliberately simplify the architecture, data distribution, and Attention parametrization. Other effects, including LayerNorm, embeddings, depth, optimizer dynamics, and dataset-specific structure, may also contribute in larger settings. The theory isolates one possible obstruction, not the full explanation. We believe a more general theory is possible and of great interest, perhaps incorporating also empirical evidence from other aggregation strategies, including a CLS token. As a first step, App. C.4 shows empirically that replacing mean pooling with a learned CLS token does not remove the from-scratch bottleneck on CIFAR10, and that SPT still yields a comparable gain there.

Acknowledgments

Antonio Orvieto acknowledges the financial support of the Hector Foundation and the AI2050 Early Career Fellowship from Schmidt Sciences. Coser Omar is a Ph.D. student enrolled in the National Ph.D. in Artificial Intelligence, XXXVIII cycle, course on Health and life sciences, organized by Università Campus Bio-Medico di Roma.

References

Carmen Amo Alonso, Jerome Sieber, and Melanie N Zeilinger. State space models as foundation models: A control theoretic overview. In *2025 American Control Conference (ACC)*, pages 146–153. IEEE, 2025. 1

Ido Amos, Jonathan Berant, and Ankit Gupta. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 3, 4, 5, 13, 21, 31, 32, 33, 36

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020. 2

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024. 1

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *arXiv preprint arXiv:1910.13051*, 2019. 3, 15, 36

Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021. 2, 15

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024. 1

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022a. 1

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b. 1

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 4

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962. 4

Kundan Krishna, Saurabh Garg, Jeffrey P Bigham, and Zachary Lipton. Downstream datasets make surprisingly good pretraining corpora. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023. 2

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 13, 36

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*, 2023. 8

Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems*, 2018. 13, 36

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5):5, 2017. 3

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, 2011. URL https://www.aclweb.org/anthology/P11-1015. 13, 36

Nikita Nangia and Samuel R Bowman. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018. 13, 36

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022. 8

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, 2023. 1

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, et al. Eagle and Finch: RWKV with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024. 1

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025. 1

Junyi Peng, Oldřich Plchot, Themos Stafylakis, Ladislav Mošner, Lukáš Burget, and Jan Černockỳ. Improving speaker verification with self-pretrained transformer models. *arXiv preprint arXiv:2305.10517*, 2023. 2

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023. 1

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, et al. Mechanistic design and scaling of hybrid architectures. *arXiv preprint arXiv:2403.17844*, 2024. 1

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021. 8

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. *arXiv preprint arXiv:2404.07904*, 2024. 1

Dragomir R Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. The acl anthology network corpus. *Language Resources and Evaluation*, 47:919–944, 2013. 13, 36

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023. 1

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 8

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020. 1, 2, 36

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, 2021. 5

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of Mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024. 1

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020. 3

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024a. 1

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024b. 1

Lei Zhou, Huidong Liu, Joseph Bae, Junjun He, Dimitris Samaras, and Prateek Prasanna. Self pre-training with masked autoencoders for medical image classification and segmentation. In *2023 IEEE 20th international symposium on biomedical imaging (ISBI)*, pages 1–6. IEEE, 2023. 2

Itamar Zimerman and Lior Wolf. On the long range abilities of transformers. *arXiv preprint arXiv:2311.16620*, 2023. 1, 2, 8, 9

A Setup

We describe below the LRA tasks, models, and experimental details.

A.1 Tasks

- **ListOps:** Inspired by [Nangia and 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 CIFAR10 [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.

A.2 Experimental details

Our training settings in most ablations closely follow choices by Amos et al. [2024].

- For **ListOps**, the model is trained with an embedding size of 512, passed through 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. Mean 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 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 the number of layers and datasets, we found these settings to be mostly stable, with only minor adjustments to the learning rate needed.

A.3 Model

We always use a pre-norm Transformer model. Given a minibatch

$$\mathcal{B} = \{(s_b, y_b)\}_{b=1}^{B}, \qquad s_b = (s_{b,1}, \ldots, s_{b,\ell_b}), \qquad \ell_b \leq L,$$

each input element is mapped to $X_{b,t} \in \mathbb{R}^D$. For discrete or quantized inputs, $X_{b,t} = E(s_{b,t})$; for continuous inputs, $X_{b,t} = E(x_{b,t}^{\text{raw}})$, where E is a learned linear encoder. Sinusoidal positional encodings are added once at the input:

$$Z_{b,t}^{(0)} = \text{Drop}(X_{b,t} + P_t).$$

For layer $\ell = 0, \ldots, T-1$, we write $Z = Z^{(\ell)}$ and $\widetilde{Z} = \text{LN}(Z)$. With H heads and $d = D/H$, the head projections are

$$W_Q^{\ell,h}, W_K^{\ell,h}, W_V^{\ell,h} \in \mathbb{R}^{d \times D},$$

and

$$Q_h = \widetilde{Z}(W_Q^{\ell,h})^\top, \qquad K_h = \widetilde{Z}(W_K^{\ell,h})^\top, \qquad V_h = \widetilde{Z}(W_V^{\ell,h})^\top.$$

Hence the raw score matrix for head h is

$$S_h = \frac{Q_h K_h^\top}{\sqrt{d}} = \frac{\widetilde{Z}(W_Q^{\ell,h})^\top W_K^{\ell,h} \widetilde{Z}^\top}{\sqrt{d}}.$$

We therefore interpret $(W_Q^{\ell,h})^\top W_K^{\ell,h} \in \mathbb{R}^{D \times D}$. The row-wise attention weights and multi-head output are

$$A_h = \text{softmax}_{\text{row}}(S_h), \qquad \text{MHA}_\ell(\widetilde{Z}) = \text{Concat}_{h=1}^H(A_h V_h) W_O^\ell.$$

The residual updates are

$$Z' = Z + \text{Drop}(\text{MHA}_\ell(\text{LN}(Z))),$$
$$Z^{(\ell+1)} = Z' + \text{Drop}(\text{MLP}_\ell(\text{LN}(Z'))),$$

where

$$\text{MLP}_\ell(x) = W_2^\ell \sigma(W_1^\ell x + b_1^\ell) + b_2^\ell.$$

For supervised classification, the final sequence is mean-pooled over non-padding positions:

$$h_b = \frac{1}{\ell_b} \sum_{t=1}^{\ell_b} Z_{b,t,:}^{(T)}, \qquad \hat{y}_b = W_{\text{cls}} h_b + b_{\text{cls}},$$

and the model is trained with cross-entropy against y_b.

For self-pretraining, a mask set $\mathcal{M}_b \subseteq \{1, \ldots, \ell_b\}$ is sampled for each sequence. The encoder receives the masked input, while the target is the original unmasked input element

$$u_{b,t} = \begin{cases} x_{b,t}^{\text{raw}}, & \text{continuous inputs,} \\ s_{b,t}, & \text{discrete or quantized inputs.} \end{cases}$$

A token-wise head predicts

$$r_{b,t} = W_{\text{spt}} Z_{b,t,:}^{(T)} + b_{\text{spt}},$$

where the output dimension is the raw input dimension for continuous reconstruction and the vocabulary or quantization size for discrete reconstruction. The SPT objective is

$$\mathcal{L}_{\text{SPT}} = \frac{1}{\sum_b |\mathcal{M}_b|} \sum_{b=1}^B \sum_{t \in \mathcal{M}_b} \ell_{\text{SPT}}^{(m)}(r_{b,t}, u_{b,t}),$$

with

$$\ell_{\text{SPT}}^{(m)}(r, u) = \begin{cases} \frac{1}{2}\|r - u\|_2^2, & \text{continuous targets,} \\ -\log \text{softmax}(r)_u, & \text{discrete or quantized targets.} \end{cases}$$

Thus SPT reconstructs the original input element, not the embedded vector $X_{b,t}$.

B Further Experimental results

B.1 SPT evidence beyond the LRA

We chose four datasets in Dempster et al. [2019] to showcase that SPT also works beyond the LRA, as suggested from earlier investigations [El-Nouby et al., 2021]. GunPoint and GunPointAgeSpan are hand-motion datasets. ECG200 and ECGFiveDays are electrocardiogram datasets. We ran each experiment on five random seeds, reporting mean plus standard deviation.

Table 6: Validation accuracy on UCR datasets: training from scratch vs. self-pretraining (SPT). Higher is better; bold indicates the best result per row.

Dataset	**From Scratch**	**SPT**
GunPoint	0.914 ± 0.024	0.964 ± 0.013
GunPointAgeSpan	0.923 ± 0.026	0.955 ± 0.011
ECG200	0.919 ± 0.019	0.954 ± 0.030
ECG5Days	0.926 ± 0.008	0.931 ± 0.011
Average	0.921 ± 0.019	0.951 ± 0.016

B.2 Precise numerical results for Fig. 2

Table 7: Ablation on number of epochs, see § 3.1. These data are shown in Fig. 2

Pretr. Len.	ListOps	CIFAR10	PathFinder
0 epochs	0.402	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)

B.3 Precise numerical results for Fig. 3

Table 8: Ablation over the number of layers; numerical values for Fig. 3.

Data	**Modality**	**1 layer**	**2 layers**	**3/4 layers**
CIFAR 10	From scratch (train)	0.69	0.78	0.88
	From scratch (val)	0.48	0.48	0.50
	+ Self Pre-trained (train)	0.93	0.95	0.96
	+ Self Pre-trained (val)	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 (val)	0.56 (+0.08)	0.56 (+0.08)	0.57 (+0.07)
PathFinder	From scratch (train)	0.67	0.76	0.83
	From scratch (val)	0.669	0.670	0.672
	Self Pre-trained (train)	0.77	0.85	0.97
	Self Pre-trained (val)	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 (val)	0.68 (+0.01)	0.68 (+0.01)	0.68 (+0.01)

B.4 Full Ablation on CIFAR and PathFinder 1 Layer

The minimal setup needed to recover full SPT initialization performance in 1 layer is to initialize only the query and key (QK) projections. On CIFAR10, QK+Enc+Norm performs best, improving

over full initialization by about +3 percentage points, while Att+Enc+Norm yields a smaller gain. Most other strategies provide only marginal improvements, except Att+MLP+Norm, which matches the Only Attention configuration. On PathFinder, the trend is similar but more nuanced: QK+Norm, Att+MLP+Norm, and Att+Enc+Norm slightly surpass full initialization, while Only QK nearly matches it. Most remaining configurations offer little benefit over training from scratch.

Table 9: One-layer models (extended Tb. 4): ablation of SPT initialization strategies focusing on Attention and QK parameters. See full discussion in §3.4.

Dataset	Full Init	Only Att	Only QK	QK+Enc+Norm	Att+Enc+Norm	QK+Norm	From Scratch	Lin+Att+MLP	Att+MLP+Norm
CIFAR10	0.603	0.598	0.607	0.64	0.62	0.521	0.486	0.540	0.598
PathFinder	0.705	0.699	0.704	0.701	0.708	0.7105	0.672	0.687	0.71

B.5 Weights Analysis

In the following tables, R denotes random initialization, SPT denotes the checkpoint after masked reconstruction, SC denotes the checkpoint obtained by supervised training from scratch, and FT denotes the checkpoint obtained by finetuning the SPT model on labels. For a parameter block θ, each entry $A \to B$ reports the Frobenius displacement $\|\theta_B - \theta_A\|_F$, while the final columns report the corresponding parameter norms. These measurements separate actual training trajectories, such as $R \to SC$, $R \to SPT$, and $SPT \to FT$, from endpoint comparisons such as $SC \to FT$. They are intended to quantify whether supervised training from random initialization substantially reorganizes the same parameter blocks that are reshaped by the SPT pipeline.

Let $Q_c^{(s)}$ denote the (flattened) weight tensor of component c at training stage $s \in \{\mathrm{R}, \mathrm{SPT}, \mathrm{SC}, \mathrm{FT}\}$. We report the parameter displacement $\|Q_c^{(s_2)} - Q_c^{(s_1)}\|$ between stages and the parameter norm $\|Q_c^{(s)}\|$, both computed as ℓ_2 (Frobenius) norms.

B.5.1 Analysis for CIFAR

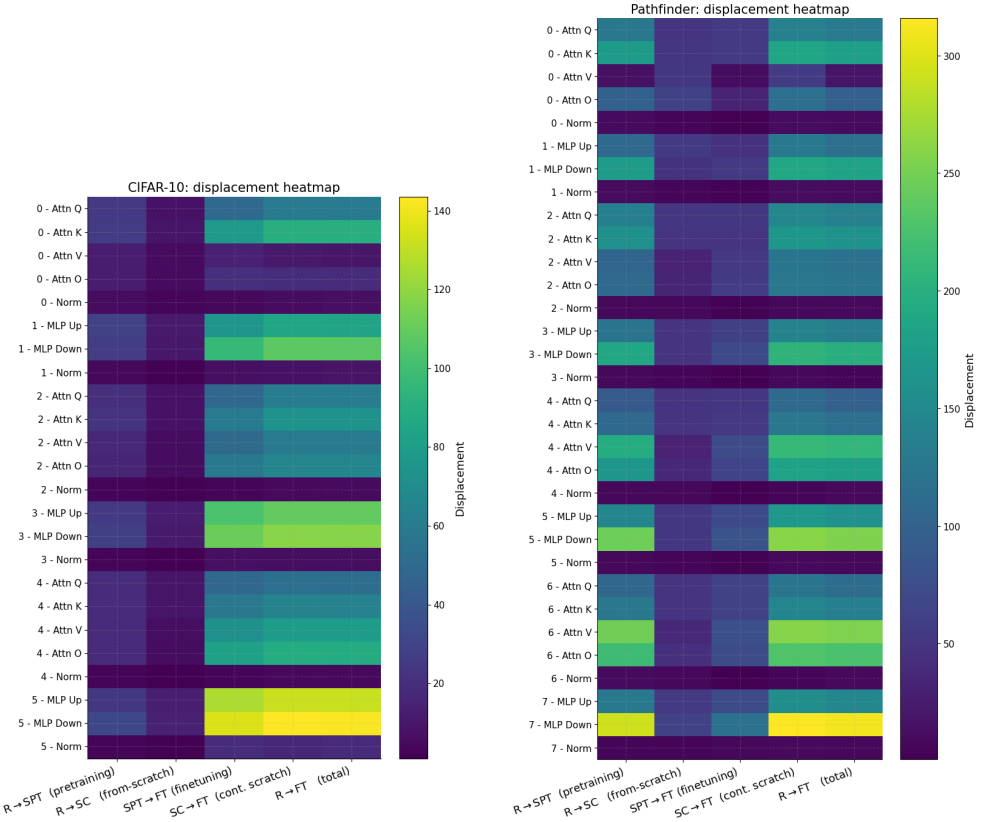
Table 10: Encoder input parameter displacement and norms across training stages. The top row reports the aggregate encoder statistics, while subsequent rows provide a layer- and component-wise breakdown. The difference between the weight norms obtained from training from scratch and random initialization is small for the encoder ($\Delta_{\text{norm}} = \|Q_{SC}\| - \|Q_R\| = -0.25$), indicating that training from scratch does not significantly alter the scale of the encoder input parameters relative to random initialization.

Layer	Component	R→SPT	R→SC	SPT→FT	SC→FT	R→FT	$\|Q_R\|$	$\|Q_{SPT}\|$	$\|Q_{SC}\|$	$\|Q_{FT}\|$
Encoder		7.94	0.66	12.22	18.64	18.75	4.39	5.84	4.14	17.34
0	Attn Q	24.51	7.90	49.55	59.50	59.43	4.66	25.48	9.51	60.22
0	Attn K	25.87	8.86	77.88	90.54	90.73	4.66	26.98	10.31	91.58
0	Attn V	11.61	5.35	14.16	10.98	9.44	4.66	12.07	7.32	8.20
0	Attn O	12.16	5.08	20.15	19.81	19.26	4.66	12.01	6.35	18.93
0	Norm	5.76	2.71	3.84	5.87	6.75	8.00	7.20	6.95	4.80
1	MLP Up	28.41	11.01	76.04	84.73	84.34	6.60	29.30	13.44	84.98
1	MLP Down	26.96	9.98	96.68	107.65	107.17	4.62	27.38	11.36	107.41
1	MLP All	39.17	14.86	123.00	136.99	136.38	8.05	40.10	17.60	136.97
1	Norm	4.05	1.59	6.86	8.19	8.63	8.00	6.64	7.61	11.24
2	Attn Q	20.61	7.79	49.30	60.86	60.76	4.67	21.68	9.46	61.84
2	Attn K	22.47	8.07	59.87	74.09	74.11	4.67	23.65	9.85	75.14
2	Attn V	17.56	5.97	50.13	59.42	59.18	4.67	18.06	7.86	59.67
2	Attn O	15.83	5.58	58.62	65.75	65.54	4.67	16.35	7.51	66.03
2	Norm	3.56	1.77	3.35	4.57	5.34	8.00	6.34	7.17	4.56
3	MLP Up	24.95	11.82	102.90	109.50	109.44	6.63	25.25	14.24	109.73
3	MLP Down	27.74	11.46	111.26	118.17	117.38	4.65	27.87	12.85	117.45
3	MLP All	37.31	16.46	151.55	161.10	160.49	8.10	37.60	19.18	160.73
3	Norm	3.62	1.30	6.96	6.26	6.18	8.00	6.19	8.33	11.70
4	Attn Q	18.96	9.08	48.32	53.14	52.66	4.73	19.88	10.67	53.24
4	Attn K	19.16	8.84	57.86	64.84	64.36	4.73	20.23	10.53	64.99
4	Attn V	18.59	6.30	72.57	79.92	79.89	4.73	19.36	8.30	80.34
4	Attn O	18.60	5.60	82.62	89.12	88.90	4.73	19.47	7.90	89.39
4	Norm	3.07	0.97	3.66	4.53	4.65	8.00	7.50	7.85	4.50
5	MLP Up	24.12	12.93	125.84	131.78	131.52	6.58	24.53	15.03	131.73
5	MLP Down	31.98	13.98	135.84	143.46	142.55	4.64	32.28	15.15	142.70
5	MLP All	40.06	19.04	185.17	194.80	193.96	8.05	40.54	21.34	194.21
5	Norm	2.88	1.63	18.84	17.62	18.48	8.00	7.12	8.96	25.47

B.5.2 Analysis for PathFinder

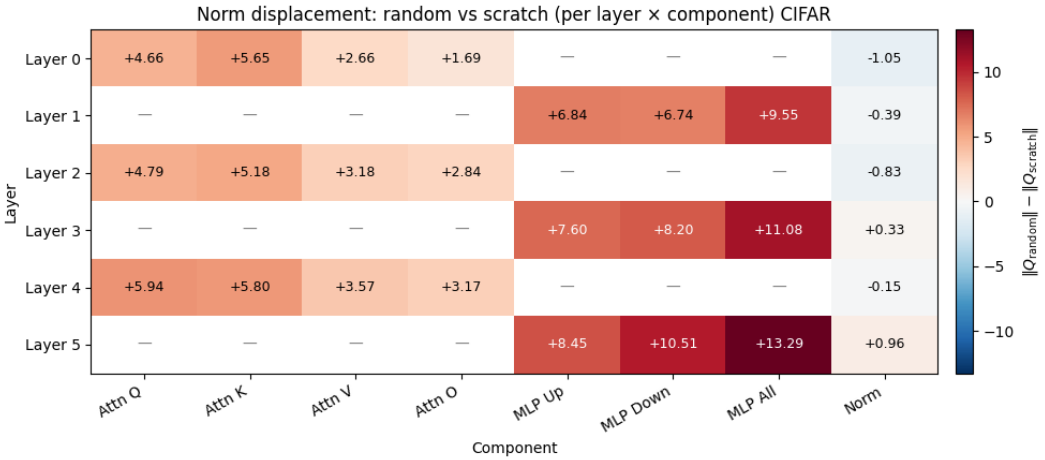
Table 11: Encoder input parameter displacement and norms across training stages. The top row reports the aggregate encoder statistics, while subsequent rows provide a layer- and component-wise breakdown. The difference between the weight norms obtained from training from scratch and random initialization is small for the encoder ($\Delta_{\text{norm}} = \|Q_{SC}\| - \|Q_R\| = -0.16$), indicating that training from scratch does not significantly alter the scale of the encoder input parameters relative to random initialization.

Layer	Component	R→SPT	R→SC	SPT→FT	SC→FT	R→FT	$\|Q_R\|$	$\|Q_{SPT}\|$	$\|Q_{SC}\|$	$\|Q_{FT}\|$
Encoder		17.96	3.57	1.85	19.47	18.92	6.70	16.90	6.54	18.05
0	Attn Q	127.64	48.66	54.24	138.04	129.23	6.57	127.76	49.11	129.32
0	Attn K	172.04	50.52	54.30	185.61	178.34	6.57	172.33	50.99	178.60
0	Attn V	15.42	52.44	11.64	55.59	18.32	6.57	13.99	53.05	17.15
0	Attn O	98.96	62.65	32.75	116.54	97.86	6.57	98.83	63.49	97.73
0	Norm	10.69	6.69	0.94	11.23	10.59	11.31	4.70	11.86	4.34
1	MLP Up	108.61	53.88	46.07	128.11	116.72	6.57	108.93	54.81	117.05
1	MLP Down	173.71	47.34	54.66	189.87	184.05	6.59	174.00	48.07	184.33
1	MLP All	204.87	71.73	71.48	229.05	217.95	9.30	205.28	72.90	218.36
1	Norm	10.30	5.39	3.68	10.90	9.93	11.31	9.45	11.96	11.07
2	Attn Q	137.44	50.44	51.81	146.12	137.71	6.57	137.67	50.85	137.94
2	Attn K	159.23	48.64	49.88	167.84	160.89	6.57	159.67	49.10	161.34
2	Attn V	103.28	31.03	53.13	123.11	119.14	6.57	103.49	31.64	119.41
2	Attn O	108.03	33.27	57.09	126.12	122.11	6.57	108.13	33.79	122.23
2	Norm	7.82	7.31	2.68	7.65	8.52	11.31	8.35	8.03	6.86
3	MLP Up	121.88	48.94	60.51	141.23	133.06	6.54	121.89	49.51	133.12
3	MLP Down	189.59	49.10	70.81	205.75	199.57	6.60	189.76	49.59	199.72
3	MLP All	225.39	69.33	93.14	249.56	239.86	9.29	225.54	70.07	240.02
3	Norm	7.30	6.25	2.13	7.22	6.64	11.31	9.03	9.73	8.99
4	Attn Q	89.88	48.52	50.80	108.37	97.04	6.56	90.05	49.04	97.17
4	Attn K	108.07	49.87	52.84	125.93	116.10	6.56	108.47	50.24	116.48
4	Attn V	196.25	30.69	72.89	211.59	209.45	6.56	196.62	31.30	209.80
4	Attn O	166.50	35.85	64.68	183.96	179.94	6.56	167.14	36.26	180.54
4	Norm	8.27	7.85	1.40	6.22	8.73	11.31	6.10	7.32	5.42
5	MLP Up	145.63	52.05	69.95	168.30	160.38	6.54	145.63	52.47	160.44
5	MLP Down	245.78	52.32	81.35	259.95	255.42	6.57	245.92	52.77	255.57
5	MLP All	285.68	73.80	107.29	309.68	301.60	9.27	285.81	74.42	301.76
5	Norm	7.81	6.27	2.73	9.48	7.97	11.31	10.12	9.71	11.60
6	Attn Q	104.73	48.12	61.59	121.54	111.79	6.58	104.81	48.53	111.83
6	Attn K	127.40	48.75	63.64	145.94	137.34	6.58	127.44	49.16	137.38
6	Attn V	247.10	36.29	77.28	258.34	255.48	6.58	247.23	36.83	255.60
6	Attn O	217.16	44.17	73.04	229.59	225.50	6.58	217.34	44.79	225.69
6	Norm	9.59	7.45	1.44	7.04	8.84	11.31	6.47	7.10	6.73
7	MLP Up	128.53	54.64	73.37	156.92	147.30	6.57	128.75	55.06	147.45
7	MLP Down	292.75	61.59	120.01	315.85	310.40	6.56	292.94	61.97	310.53
7	MLP All	319.73	82.33	140.66	352.68	343.57	9.29	319.98	82.90	343.76
7	Norm	6.96	5.02	7.41	9.34	7.72	11.31	8.04	10.77	14.29

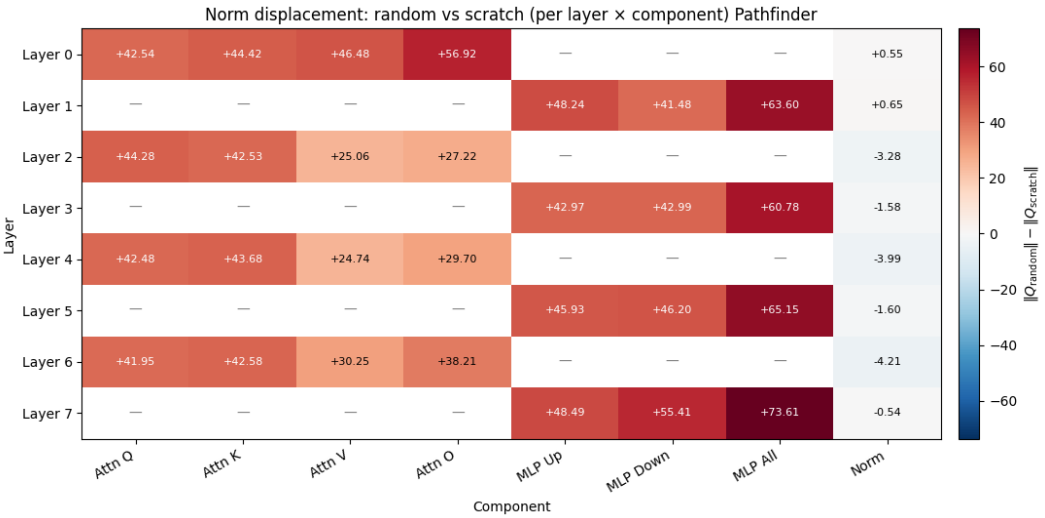


(a) CIFAR10 (b) PathFinder

Figure 8: *Layer-wise parameter displacement across training trajectories.* R→SC displacement is consistently smaller than R→SPT and SPT→FT, showing that supervised training from random initialization induces limited movement. MLP layers move substantially more than Attention projections, while Attention layers move little unless initialized from SPT.



(a) CIFAR10



(b) PathFinder

Figure 9: *Change in parameter norm from random initialization to from-scratch training,* $\|Q_{\mathrm{SC}}\| - \|Q_{\mathrm{R}}\|$*, per layer and component.* The norm change is largest for MLP blocks and generally increases with depth, while normalization layers change little. Attention projections change less than MLP blocks, in line with the displacements in Tbs. 10 and 11.

C Additional Experiments: Baselines, Compute, and Aggregation

This section collects experiments that test whether the SPT gap in § 3 could be explained by an under-tuned or under-trained supervised baseline, by the computational budget, or by the choice of mean pooling as the aggregation strategy.

C.1 Compute-matched and learning-rate-tuned from-scratch baseline

We train from scratch for 400 epochs, which exceeds the total SPT+finetuning budget used in the main text, and sweep the learning rate over four orders of magnitude around each task's base rate. We use two schedules: the constant-with-warmup schedule of Amos et al. [2024] and cosine decay with warmup. All other settings follow App. A.2. Results are in Tb. 12.

No configuration improves over the tuned 100-epoch baseline of the main text. The best settings reach 0.501 on CIFAR10 and 0.685 on PathFinder, far below the SPT accuracies obtained with a 100-epoch finetuning budget (0.724 and 0.85). In all stable configurations, validation accuracy peaks within roughly 50 epochs and then degrades, so additional supervised epochs do not close the gap. Large learning rates (10× base) diverge or remain near chance, and very small rates on PathFinder fail to leave chance level.

Table 12: **Compute-matched from-scratch baseline** (400 epochs). Learning rates are swept around each task's base rate under two schedulers. We report mean ± standard deviation of peak validation accuracy over seeds. Best result per scheduler and dataset in bold. For reference, SPT reaches 0.724 (CIFAR10) and 0.85 (PathFinder) with 100 finetuning epochs. Tested on 3 random seeds.

Scheduler	lr scale	lr (CIFAR10)	Acc. (CIFAR10)	lr (PathFinder)	Acc. (PathFinder)
Constant + warmup	10× base	1e−2	0.278 ± 0.08	5e−3	0.50 ± 0.00
	≈ base	1e−3	0.489 ± 0.005	2.5e−4	0.675 ± 0.04
	0.1× base	1e−4	$\mathbf{0.501 \pm 0.003}$	5e−5	$\mathbf{0.683 \pm 0.05}$
	0.01× base	1e−5	$\mathbf{0.501 \pm 0.003}$	5e−6	0.50 ± 0.00
	0.001× base	1e−6	0.321 ± 0.007	5e−7	0.50 ± 0.00
Cosine + warmup	10× base	1e−2	0.272 ± 0.12	5e−3	0.50 ± 0.00
	≈ base	1e−3	$\mathbf{0.496 \pm 0.06}$	2.5e−4	0.66 ± 0.08
	0.1× base	1e−4	0.492 ± 0.03	5e−5	$\mathbf{0.685 \pm 0.07}$
	0.01× base	1e−5	0.498 ± 0.06	5e−6	0.50 ± 0.00
	0.001× base	1e−6	0.315 ± 0.08	5e−7	0.50 ± 0.00

C.2 Wall-clock cost of SPT versus extended supervised training

Tb. 13 reports wall-clock time per run on a single NVIDIA A100-80GB with batch size 16, averaged over seeds where applicable. The full SPT pipeline costs less wall-clock time than the extended 400-epoch from-scratch baseline on both tasks (2h 09m vs. 5h 40m on CIFAR10; 29h 16m vs. 32h 18m on PathFinder), while reaching substantially higher accuracy (0.724 vs. 0.501 on CIFAR10; 0.85 vs. 0.685 on PathFinder). Additional supervised budget therefore does not account for the gap in our setup.

C.3 Hyperparameter tuning for partial SPT initialization

To check whether the gap between partial and full SPT initialization in Tb. 3 is an artifact of hyperparameters, we run a dedicated search over learning rate, warmup length, and weight decay on PathFinder for the Att+Norm+Enc configuration (SPT values for the Attention blocks, their normalization layers, and the encoder; all other parameters randomly initialized). Results are in Tb. 14. Partial initialization recovers part of the gap (up to 0.781), but no combination approaches full SPT initialization (0.85). Hyperparameter tuning thus does not substitute for full SPT initialization, either from scratch (App. C.1) or with partial initialization.

Table 13: **Wall-clock training time and accuracy** (single A100-80GB, batch size 16). †SPT pipeline: 100 pretraining + 100 finetuning epochs on CIFAR10; 200 pretraining + 100 finetuning epochs on PathFinder.

Regime	Epochs	Time (CIFAR10)	Time (PathFinder)	Acc. (CIFAR10)	Acc. (PathFinder)
Pretraining (MLM)	100	1h 10m	9h 39m	–	–
Pretraining (MLM)	200	2h 38m	19h 21m	–	–
Pretraining (MLM)	400	5h 10m	39h 09m	–	–
Finetuning (from SPT)	100	0h 59m	9h 55m	0.724	0.85
From scratch	100	0h 59m	8h 45m	0.503	0.672
SPT pipeline†	PT+FT	**2h 09m**	**29h 16m**	**0.724**	**0.85**
From scratch (ext.)	400	**5h 40m**	**32h 18m**	**0.501**	**0.685**

Table 14: **Hyperparameter search for partial SPT initialization** (Att+Norm+Enc) on PathFinder.

Learning rate	Warmup steps	Weight decay	Accuracy
2e−4	2,000	0.01	0.707
3e−4	2,000	0.01	0.702
4e−4	3,000	0.0	0.700
3.5e−4	3,000	0.0	**0.778**
2.5e−4	3,000	0.0	0.751
From scratch (reference)			*0.672*
Full SPT init (reference)			*0.85*

C.4 CLS token aggregation

All main-text experiments use mean pooling. To test whether the SPT gap is specific to this aggregation, we replace mean pooling with a learned CLS token and compare training from scratch against finetuning from the same SPT checkpoint (5 seeds per cell). We also sweep the learning rate for the CLS configuration, so that results are not an artifact of a learning rate tuned for mean pooling. Results are in Tb. 15.

On CIFAR10, the from-scratch model performs comparably poorly with either readout (0.483–0.484 with CLS vs. 0.503 with mean pooling), and the result is stable across the learning-rate sweep. SPT recovers a comparable gain with CLS ($+0.17$) and with mean pooling ($+0.22$), so the from-scratch bottleneck and the benefit of SPT carry over across aggregation schemes on this task.

On PathFinder, the CLS readout is considerably more fragile. At the learning rate used in the main text, it fails to leave chance both from scratch and from the SPT checkpoint, so SPT-initialized Attention does not rescue it: in this regime, the binding constraint is the readout rather than the Attention patterns. At a $10\times$ smaller learning rate the CLS model does train (0.623 ± 0.06 from scratch), but remains below mean pooling (0.672 from scratch, 0.85 with SPT) and shows much higher seed variance. Mean pooling is therefore the stronger and more robust readout on both tasks, which supports its use in the main pipeline.

Table 15: **Mean pooling vs. CLS token** aggregation. Mean $\pm$ standard deviation over 5 seeds. "–": not run.

	lr	From scratch	SPT
CIFAR10			
Mean pooling (main text)	1e−3	0.503 ± 0.002	0.721 ± 0.012
CLS token	1e−3	0.483 ± 0.004	0.653 ± 0.007
CLS token	1e−4	0.484 ± 0.005	–
CLS token	1e−5	0.438 ± 0.008	–
PathFinder			
Mean pooling (main text)	5e−4	0.672 ± 0.003	0.85 ± 0.004
CLS token	5e−4	0.50 ± 0.00	0.50 ± 0.00
CLS token	5e−5	0.623 ± 0.06	–
CLS token	5e−6	0.50 ± 0.00	–

D Synthetic Task

We first describe the synthetic two-dimensional binary time-series classification dataset used in § 4, and then present additional plots supporting our claims.

D.1 Task Description

Each sample is a sequence $x = (x_j)_{j=0}^{L-1} \in \mathbb{R}^{L\times 2}$ with $L = 100$ and a binary label $y \in \{0, 1\}$. With

$$a_0 = (0.66,\ 0.55), \quad b_0 = (-0.495,\ -0.605), \quad a_1 = (0.605,\ -0.55), \quad b_1 = (-0.55,\ 0.605).$$

Conditioned on the binary label, we define anchor points

$$(c_1, c_2) = \begin{cases} (a_0, b_0), & y = 0, \\ (a_1, b_1), & y = 1. \end{cases}$$

For each sequence, we sample independently

$$s \sim \mathcal{U}(0.12, 0.28), \qquad \phi \sim \mathcal{U}(0, 2\pi), \qquad \alpha \sim \mathcal{N}(1, 0.1^2).$$

Define normalized time $t_j = j/(L-1)$ and a "warped" time variable

$$\tilde{t}_j = t_j + s\sin(2\pi t_j)\big(0.5 + 0.5\cos(2\pi t_j)\big).$$

The mixing coefficient and mean trajectory are defined as

$$m_j = 0.5 + 0.35\sin(2\pi\tilde{t}_j + \phi), \qquad \mu_j = m_j c_1 + (1 - m_j)c_2.$$

Finally, a smooth oscillatory drift is added: $d_j = 0.30\big(\sin(2\pi\tilde{t}_j),\ \cos(2\pi\tilde{t}_j)\big)$. Observations are generated as

$$x_j = \alpha(\mu_j + d_j) + \varepsilon_j + \delta_j, \qquad \varepsilon_j \sim \mathcal{N}(0, \sigma^2 I_2),\ \sigma = 0.55.$$

To introduce outliers, sample $N_{\text{sp}} \sim \text{Unif}\{0, 1, 2, 3\}$ and choose a random index set $I \subset \{0, \dots, L-1\}$ with $|I| = N_{\text{sp}}$. For $j \in I$, draw $\delta_j \sim \mathcal{N}(0, I_2)$; otherwise set $\delta_j = 0$.

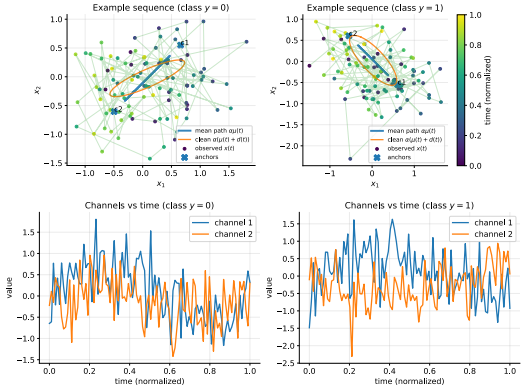


Figure 10: Illustration of the toy task for a random seed.

D.2 Additional Plots for § 4

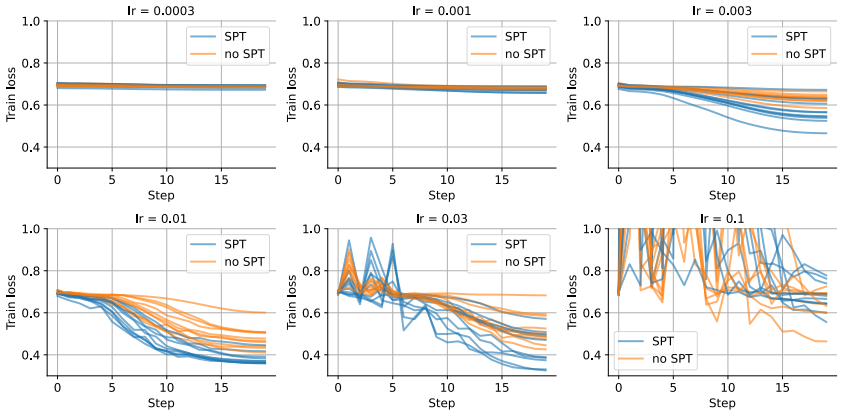


Figure 11: Complementing Fig. 4 with loss evolution over time. SPT initialization leads to faster and more stable convergence across seeds, particularly at intermediate learning rates, while differences diminish at very low learning rates and training becomes unstable at high learning rates.

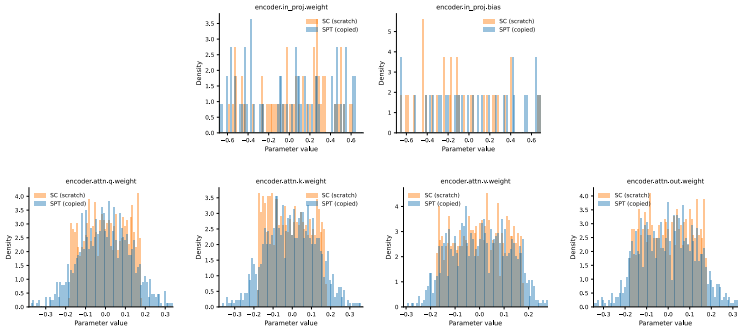


Figure 12: Weight distributions after SPT remain close to random initialization, except for mild broadening toward a Gaussian-like shape. Thus, the main structure is not visible in the marginal distributions of W_Q or W_K, but in their product and its interaction with positional encodings.

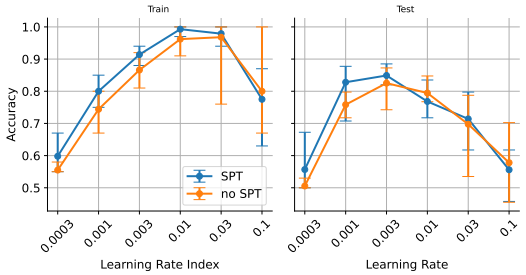


Figure 13: Same setting as Fig. 4, but finetuning is performed for 80 epochs instead of 20. SPT initialization still achieves higher accuracy across most learning rates, although the gap narrows relative to Fig. 4.

E Proofs

Here we consider the attention mechanism

$$a_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{L} \exp(s_{ik})},$$

where s_{ij} are scores that define the attention pattern. For simplicity, we consider them independent of the input. Further, we consider learning a target pattern $s^*_{ij} = \Delta_{ij}$.

We first show in the next result that we can, without loss of generality, assume that Δ is centered.

Lemma 1 (Row-centering is without loss of generality)**.** *Let $\Delta \in \mathbb{R}^{L\times L}$ be any score direction, and define its row-centered version*

$$\bar{\Delta}_{ij} := \Delta_{ij} - \frac{1}{L}\sum_{k=1}^{L} \Delta_{ik}, \qquad \textit{which implies} \qquad \bar{\Delta}\mathbf{1} = 0.$$

For any base scores b_{ij} and any $\alpha \in \mathbb{R}$, the softmax attention weights generated by

$$s^{\Delta}_{ij}(\alpha) = b_{ij} + \alpha\Delta_{ij}$$

are identical to those generated by

$$s^{\bar{\Delta}}_{ij}(\alpha) = b_{ij} + \alpha\bar{\Delta}_{ij}.$$

Consequently, any forward pass and any loss depending on the scores only through softmax attention are identical for Δ and $\bar{\Delta}$.

Proof. First, we quickly verify that centering implies $\bar{\Delta}\mathbf{1} = 0$

$$\sum_{j=1}^{L} \bar{\Delta}_{ij} = \sum_{j=1}^{L} \Delta_{ij} - \sum_{j=1}^{L} \frac{1}{L}\sum_{k=1}^{L} \Delta_{ik} = \sum_{j=1}^{L} \Delta_{ij} - \sum_{k=1}^{L} \Delta_{ik} = 0.$$

Next, let $c_i := \frac{1}{L}\sum_{k=1}^{L} \Delta_{ik}$. Then $\bar{\Delta}_{ij} = \Delta_{ij} - c_i$, so

$$s^{\bar{\Delta}}_{ij}(\alpha) = b_{ij} + \alpha\bar{\Delta}_{ij} = b_{ij} + \alpha(\Delta_{ij} - c_i) = s^{\Delta}_{ij}(\alpha) - \alpha c_i$$

For each fixed row i, the quantity αc_i is independent of j. Therefore

$$\frac{\exp(s^{\bar{\Delta}}_{ij}(\alpha))}{\sum_{k=1}^{L}\exp(s^{\bar{\Delta}}_{ik}(\alpha))} = \frac{\exp(s^{\Delta}_{ij}(\alpha) - \alpha c_i)}{\sum_{k=1}^{L}\exp(s^{\Delta}_{ik}(\alpha) - \alpha c_i)} = \frac{e^{-\alpha c_i}\exp(s^{\Delta}_{ij}(\alpha))}{e^{-\alpha c_i}\sum_{k=1}^{L}\exp(s^{\Delta}_{ik}(\alpha))} = \frac{\exp(s^{\Delta}_{ij}(\alpha))}{\sum_{k=1}^{L}\exp(s^{\Delta}_{ik}(\alpha))}.$$

Thus the attention weights are identical for all α, and so are the resulting outputs and losses. □

In the following proposition, we consider optimization of the attention matrix towards a normalized latent score Δ through a single parameter α.

Restated Proposition 1 Let $X_1, \ldots, X_L \in \mathbb{R}^d$ be centered random tokens with finite second moments, and define the autocorrelation matrix $C \in \mathbb{R}^{L\times L}$ as $C_{ij} := \mathbb{E}[X_i^\top X_j]$. For a score pattern $\Delta \in \mathbb{R}^{L\times L}$, consider training a single parameter α controlling the attention score:

$$s_{ij}(\alpha) = \alpha\Delta_{ij}, \qquad a_{ij}(\alpha) = \frac{\exp(s_{ij}(\alpha))}{\sum_{k=1}^{L}\exp(s_{ik}(\alpha))},$$

with outputs $o_i(\alpha) = \sum_j a_{ij}(\alpha)X_j$ and mean-pooled representation $h(\alpha) = L^{-1}\sum_i o_i(\alpha)$.

Consider the following losses and assume they are regular enough so that one can interchange expectation and differential computation:

$$\mathcal{L}_{\text{sup}}(\alpha) = \mathbb{E}[\ell(h(\alpha), Y)] \qquad \text{and} \qquad \mathcal{L}_{\text{SPT}}(\alpha) = \frac{1}{2L}\sum_i \mathbb{E}[\|X_i - o_i(\alpha)\|^2].$$

Let $\mathbf{1} \in \mathbb{R}^L$ be the all-ones vector and define $\mathcal{B} := \{\Gamma \in \mathbb{R}^{L\times L} : \Gamma^\top\mathbf{1} = 0\}$. Further, define $\langle A, B\rangle_F = \sum_{i,j} A_{ij}B_{ij}$ and $\bar{\Delta} = \Delta - \frac{1}{L}\Delta\mathbf{1}\mathbf{1}^\top$. For every $\Delta \in \mathcal{B}$, we have:

$$\mathcal{L}'_{\text{sup}}(0) = 0, \qquad \mathcal{L}'_{\text{SPT}}(0) = -\frac{1}{L^2}\langle\bar{\Delta}, C\rangle_F$$

Proof. Fix $\Delta \in \mathcal{B}$. Thanks to Lemma 1, without loss of generality, we can assume $\Delta \in \tilde{\mathcal{B}}$

$$\tilde{\mathcal{B}} := \{\Gamma \in \mathbb{R}^{L\times L} : \Gamma\mathbf{1} = 0,\ \Gamma^\top\mathbf{1} = 0\}.$$

First compute the derivatives of the attention weights. For each fixed row i,

$$a_{ij}(\alpha) = \frac{e^{\alpha\Delta_{ij}}}{\sum_k e^{\alpha\Delta_{ik}}}.$$

Therefore

$$a'_{ij}(\alpha) = a_{ij}(\alpha)\left(\Delta_{ij} - \sum_k a_{ik}(\alpha)\Delta_{ik}\right).$$

At $\alpha = 0$, $a_{ij}(0) = 1/L$, so

$$a'_{ij}(0) = \frac{1}{L}\left(\Delta_{ij} - \frac{1}{L}\sum_k \Delta_{ik}\right).$$

Since $\Delta\mathbf{1} = 0$, this simplifies to

$$a'_{ij}(0) = \frac{\Delta_{ij}}{L}.$$

Thus the derivative of the attention output is

$$o'_i(0) = \sum_j a'_{ij}(0)X_j = \frac{1}{L}\sum_j \Delta_{ij}X_j.$$

The derivative of the mean-pooled representation is then

$$h'(0) = \frac{1}{L}\sum_i o'_i(0) = \frac{1}{L^2}\sum_i\sum_j \Delta_{ij}X_j = \frac{1}{L^2}\sum_j\left(\sum_i \Delta_{ij}\right)X_j.$$

Since $\Delta^\top\mathbf{1} = 0$, we have $h'(0) = 0$. Hence

$$\mathcal{L}'_{\text{sup}}(0) = \mathbb{E}[\nabla_h \ell(h(0), Y)^\top h'(0)] = 0.$$

Now consider SPT. At $\alpha = 0$, all rows attend uniformly, so $o_i(0) = \bar{X} := \frac{1}{L}\sum_k X_k$. Differentiating,

$$\mathcal{L}'_{\text{SPT}}(0) = \frac{1}{L}\sum_i \mathbb{E}[(o_i(0) - X_i)^\top o'_i(0)].$$

Substituting $o_i(0) = \bar{X}$ and $o'_i(0) = L^{-1}\sum_j \Delta_{ij}X_j$,

$$\mathcal{L}'_{\text{SPT}}(0) = \frac{1}{L^2}\sum_{i,j} \Delta_{ij}\mathbb{E}[(\bar{X} - X_i)^\top X_j].$$

Let $S_j := \sum_k C_{kj}$. Since

$$\mathbb{E}[\bar{X}^\top X_j] = \frac{1}{L}\sum_k C_{kj} = \frac{S_j}{L},$$

we get

$$\mathbb{E}[(\bar{X} - X_i)^\top X_j] = \frac{S_j}{L} - C_{ij}.$$

Therefore

$$\mathcal{L}'_{\text{SPT}}(0) = \frac{1}{L^2}\sum_{i,j} \Delta_{ij}\left(\frac{S_j}{L} - C_{ij}\right).$$

The term involving S_j vanishes because $\sum_i \Delta_{ij} = 0$, and the result follows. □

Diagonal-masked variant. It is possible to show a very similar result for the case where there are token-to-same-token interactions: $a_{ii}(\alpha) = 0$ and, for $j \neq i$,

$$a_{ij}(\alpha) = \frac{e^{\alpha \Delta_{ij}}}{\sum_{k \neq i} e^{\alpha \Delta_{ik}}}.$$

Let us directly assume we work with doubly-normalized targets: $\Delta_{ii} = 0$, $\sum_{j \neq i} \Delta_{ij} = 0$, and $\sum_{i \neq j} \Delta_{ij} = 0$. Then:

$$a'_{ij}(0) = \frac{1}{L-1} \left(\Delta_{ij} - \frac{1}{L-1} \sum_{k \neq i} \Delta_{ik} \right) = \frac{\Delta_{ij}}{L-1} \qquad (j \neq i).$$

Hence $o'_i(0) = (L-1)^{-1} \sum_{j \neq i} \Delta_{ij} X_j$, and so

$$h'(0) = \frac{1}{L(L-1)} \sum_j \left(\sum_{i \neq j} \Delta_{ij} \right) X_j = 0.$$

Thus again we get $\mathcal{L}'_{\text{sup}}(0) = 0$. For SPT, $o_i(0) = \bar{X}_{-i} := (L-1)^{-1} \sum_{k \neq i} X_k$, so

$$\mathcal{L}'_{\text{SPT}}(0) = \frac{1}{L(L-1)} \sum_{i,j \neq i} \Delta_{ij} \mathbb{E}[(\bar{X}_{-i} - X_i)^\top X_j].$$

For $j \neq i$, writing $S_j := \sum_k C_{kj}$,

$$\mathbb{E}[(\bar{X}_{-i} - X_i)^\top X_j] = \frac{S_j - C_{ij}}{L-1} - C_{ij} = \frac{S_j}{L-1} - \frac{L}{L-1} C_{ij}.$$

Therefore (the first term vanishes by off-diagonal column-centering)

$$\mathcal{L}'_{\text{SPT}}(0) = \frac{1}{L(L-1)^2} \sum_j S_j \sum_{i \neq j} \Delta_{ij} - \frac{1}{(L-1)^2} \sum_{i \neq j} \Delta_{ij} C_{ij} = -\frac{1}{(L-1)^2} \sum_{i \neq j} \Delta_{ij} C_{ij}.$$

E.1 Extensions of the local cancellation argument

General token mixers. Let $u_j(X) \in \mathbb{R}^d$ be fixed encoded tokens, and suppose a perturbation of a token mixer satisfies $o'_i(0) = \sum_j \Gamma_{ij} u_j(X)$, with $\Gamma \in \mathbb{R}^{L \times L}$. For fixed aggregation weights $b \in \mathbb{R}^L$,

$$h_b(\alpha) = \sum_i b_i o_i(\alpha), \quad h'_b(0) = \sum_j (\Gamma^\top b)_j u_j(X). \tag{1}$$

Thus $\Gamma^\top b = 0$ implies a zero derivative for any differentiable supervised loss depending on α only through h_b. This covers the constrained token-mixing MLP $o(\alpha) = (W_2 + \alpha\Gamma)u(X)$, where $u(X) = \sigma(W_1 X)$ is held fixed. For the softmax Attention model of Proposition 1, $\Gamma = \Delta/L$ and mean pooling corresponds to $b = \mathbf{1}/L$. The cancellation is therefore not unique to Attention.

Input-dependent scores and multiple parameters. Let $S(\theta, X)$ be differentiable query-key scores and $a(\theta, X) = \text{softmax}_{\text{row}}(S(\theta, X))$. With value vectors $V_j(X) \in \mathbb{R}^d$, define $h(\theta, X) = L^{-1} \sum_{i,j} a_{ij}(\theta, X) V_j(X)$. Suppose θ_0 produces uniform Attention. For a query-key parameter direction v, keep the values fixed and write

$$\Gamma_v(X) = D_\theta S(\theta_0, X)[v], \quad \bar{\Gamma}_v(X) = \Gamma_v(X) - \frac{1}{L} \Gamma_v(X) \mathbf{1}\mathbf{1}^\top. \tag{2}$$

Differentiating the row-wise softmax and then mean pooling gives

$$\begin{aligned} D_\theta a_{ij}(\theta_0, X)[v] &= \frac{1}{L} \big(\bar{\Gamma}_v(X)\big)_{ij}, \\ D_\theta h(\theta_0, X)[v] &= \frac{1}{L^2} \sum_j \big(\bar{\Gamma}_v(X)^\top \mathbf{1}\big)_j V_j(X). \end{aligned} \tag{3}$$

Hence $\bar{\Gamma}_v(X)^\top \mathbf{1} = 0$ for every input is a sufficient condition for local cancellation along v, even with input-dependent scores and multiple trainable parameters.

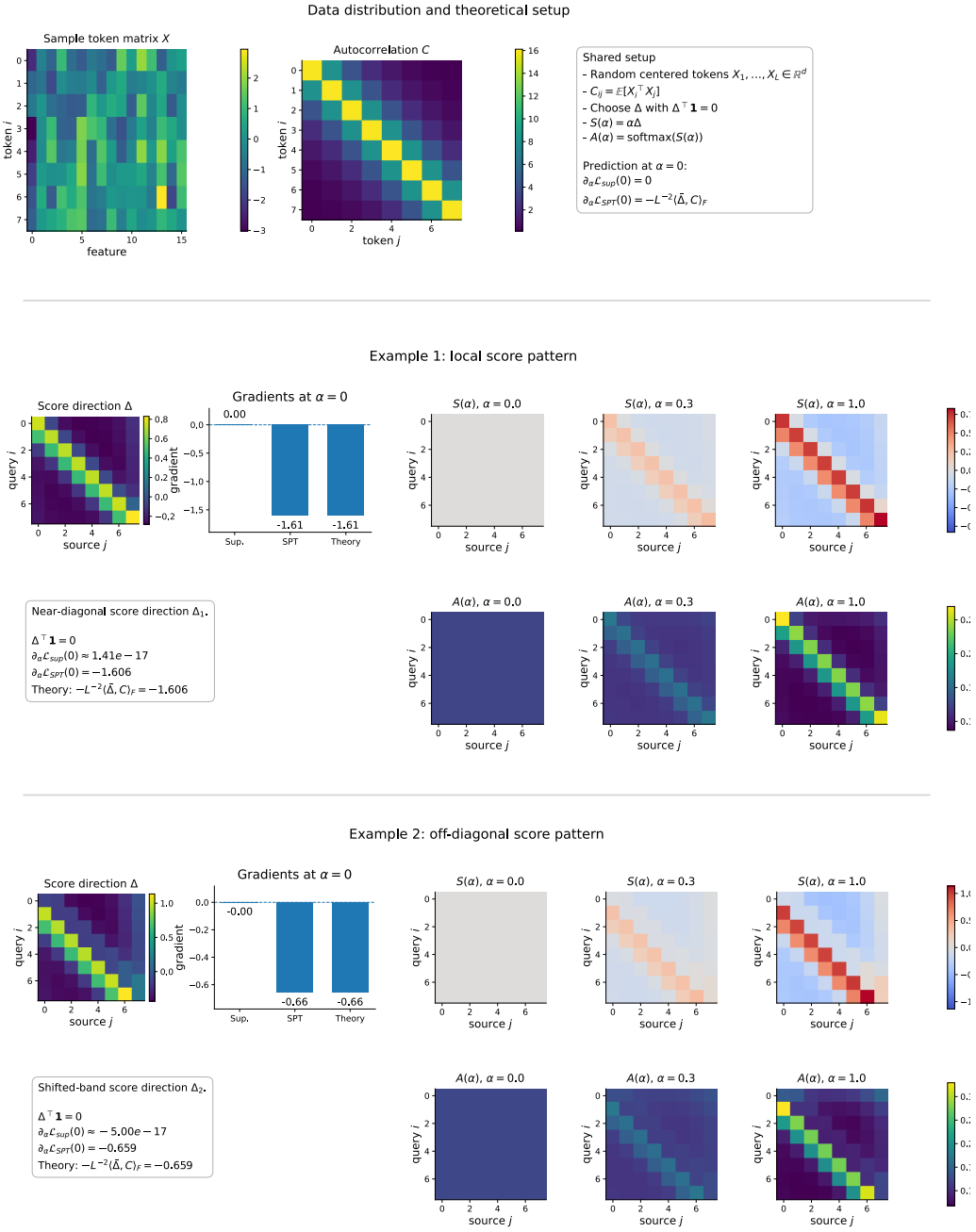


Figure 14: Verification of Proposition 1 on 2 example patterns under randomly sampled tokens.

A two-parameter example. Take $z_1 = (1, x)^\top$, $z_2 = (1, -x)^\top$, and

$$W_Q(\alpha) = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}, \quad W_K(\beta) = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix}, \quad W_V = \begin{pmatrix} 0 & 1 \end{pmatrix}. \tag{4}$$

Using unscaled scores $s_{ij} = (W_Q z_i)^\top W_K z_j$ gives

$$S(\alpha, \beta; x) = \mathbf{1}\mathbf{1}^\top + \alpha x^2 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + \beta x \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}. \tag{5}$$

The values are x and $-x$, so the mean-pooled output is $h(\alpha, \beta; x) = x(a_{11} + a_{21} - 1)$. At $(\alpha, \beta) = (0, 0)$, Attention is uniform and the preceding derivative formula yields

$$\partial_\alpha h(0, 0; x) = 0, \quad \partial_\beta h(0, 0; x) = x^2. \tag{6}$$

Thus local cancellation in one parameter can coexist with sensitivity to another.

A simplified CLS readout. Consider the non-residual Attention outputs $o_i(\alpha)$ of Proposition 1, followed by

$$h(\alpha, \beta) = \sum_i b_i(\alpha, \beta) o_i(\alpha), \quad \sum_i b_i(\alpha, \beta) = 1. \tag{7}$$

Here β controls only the final aggregation scores. At $\alpha = 0$, all incoming tokens equal $\bar{X} = L^{-1} \sum_j X_j$; additionally assume $b_i(0, 0) = 1/L$. Differentiating $\sum_i b_i = 1$ cancels the terms involving derivatives of the aggregation weights, giving

$$\begin{aligned} &\partial_\beta h(0, 0) = 0, \\ &\partial_\alpha h(0, 0) = \frac{1}{L} \sum_i o_i'(0) = \frac{1}{L^2} \sum_j (\bar{\Delta}^\top \mathbf{1})_j X_j. \end{aligned} \tag{8}$$

The same cancellation therefore holds when $\bar{\Delta}^\top \mathbf{1} = 0$. This calculation relies on identical incoming token representations; uniform CLS weights alone are insufficient.

Causal convolution kernels. For a linear SSM with zero initial state, let $K = (K_0, \ldots, K_{L-1})$ denote its causal convolution kernel. Mean pooling gives

$$h_K(X) = \frac{1}{L} \sum_{t=1}^{L} \sum_{j=1}^{t} K_{t-j} X_j = \frac{1}{L} \sum_{j=1}^{L} \left(\sum_{k=0}^{L-j} K_k \right) X_j. \tag{9}$$

If $h_K(X) = h_{K^\star}(X)$ for every input sequence, equality of the coefficients of each X_j implies

$$\sum_{k=0}^{L-j} (K_k - K_k^\star) = 0, \qquad j = 1, \ldots, L. \tag{10}$$

Taking $j = L, L-1, \ldots, 1$ successively gives $K_k = K_k^\star$ for $k = 0, \ldots, L-1$. Thus distinct causal kernels over the observed horizon cannot be indistinguishable after mean pooling for every input. This is a kernel-level distinction, not a guarantee of nonzero SSM parameter gradients or successful supervised optimization.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 2 (Preliminaries) together with the introduction state three claims that map directly onto our contributions: (i) we replicate Amos et al. [2024] (Table 1 and Section B.1); (ii) we ablate SPT along pretraining duration (Section 3.1), depth and data source (Section 3.2), weight freezing (Section 3.3), and partial initialization (Section 3.4), identifying Attention-score learning – and specifically W_Q, W_K – as the main carrier of the SPT benefit (Takeaways 1–4 in Section 3); (iii) we provide a controlled synthetic task (Section 4) and a theoretical statement (Proposition 1 in Section 5) showing that mean-pooled label supervision can be locally blind to score directions detectable by masked reconstruction. The scope is explicitly limited to sequence classification on LRA-style tasks plus a small synthetic setup; we do not claim state-of-the-art performance or universality.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated Limitations paragraph is provided at the end of Section 6. We note that (i) the most detailed multi-seed component ablations are restricted to CIFAR-10 and Pathfinder; (ii) PathX is excluded for compute reasons; (iii) several large LRA runs are single-seed, as disclosed in Section 2; (iv) the synthetic task in Section 4 and the theoretical analysis in Section 5 deliberately simplify the architecture (one layer, no MLP, no LayerNorm, input-independent scores); and (v) other factors such as LayerNorm, embeddings, depth, and optimizer dynamics may also contribute in larger settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The single theoretical result, Proposition 1, is stated formally in Section 5 with all assumptions: centered tokens with finite second moments, input-independent scores, the one-parameter family $s_{ij}(\alpha) = \alpha\Delta_{ij}$, the mean-pooled supervised loss, the masked-reconstruction SPT loss, and $\Delta \in \mathcal{B} = \{\Gamma : \Gamma^\top \mathbf{1} = 0\}$. Lemma 1 (row-centering is without loss of generality) and the full proof of Proposition 1 are given in Appendix E, including the diagonal-masked variant. A numerical verification on two example score patterns is shown in Figure 14.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2 specifies the model, training pipeline (200 epochs SPT followed by 100 epochs finetuning, vs. 100 epochs from-scratch), optimizer (AdamW), masking ratios, evaluation protocol (peak held-out accuracy on the public LRA split), and seed convention (5 seeds for CIFAR-10 and Pathfinder component ablations, single-seed for the larger sweeps, with explicit disclosure). Appendix A.1 describes the LRA tasks; Appendix A.2 lists per-dataset hyperparameters (embedding size, depth, heads, FFN size, learning rates, batch sizes, weight decay, pretraining loss); Appendix A.3 fully specifies the pre-norm Transformer and the SPT objective for both continuous and discrete inputs. Appendix D gives the full generative process for the synthetic task with all distributional parameters. We build directly on the public codebase of Amos et al. [2024] with the same Transformer settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No] Justification: We do not release separate code. All experiments are run with the public codebase of Amos et al. [2024] (https://github.com/IdoAmos/not-from-scratch), using the settings in Appendix A.2. Our modifications to it (partial SPT initialization, weight freezing, and the CLS readout) are fully described in Sections 2–3 and Appendices A and C. The synthetic task is fully specified by the generative procedure in Appendix D and can be reproduced from that description. The LRA and UCR datasets are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 2 reports the training and evaluation protocol; Appendix A.2 lists per-dataset architectures and hyperparameters (learning rates, batch sizes, weight decay, pretraining loss type, masking ratio); Appendix A.3 fully specifies the pre-norm Transformer, multi-head attention, MLP, mean pooling, and the SPT head. Appendix D documents the synthetic task. We follow the LRA convention of using the public held-out split as the evaluation split.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the experiments supporting the Attention-specific claims – freezing in Table 2, hybrid initialization in Table 3, and the one-layer QK ablation in Table 4 – we report mean $\pm$ standard deviation across 5 random seeds on CIFAR10 and PathFinder; the ListOps, Retrieval, and Text entries of Table 2 report mean $\pm$ standard deviation across 3 random seeds. The compute-matched from-scratch baseline (Table 12) reports mean $\pm$ standard deviation over 3 random seeds, and the CLS-token comparison (Table 15) over 5 random seeds. The synthetic task results (Figure 4, with additional plots in Figures 11 and 13) report mean and max–min interval over 10 seeds. The UCR experiments in Appendix B.1 (Table 6) report mean $\pm$ standard deviation over 5 seeds. As stated in Section 2, the remaining larger LRA sweeps (e.g., Tables 1, 5, 7, 8, 9, and 14) are single-seed runs due to compute; this limitation is explicitly disclosed, and we observed low variation across seeds in the maximum accuracy on CIFAR10 and PathFinder, which justified the choice. Error bars are 1-sigma standard deviations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were run on a single NVIDIA A100 GPU (80 GB). Per-epoch wall-clock times are reported in the footnote at the start of Section 3.1: roughly 48 s for CIFAR pretraining, 19.1 min for ListOps pretraining, and 3.20 min for Pathfinder pretraining; finetuning takes 56 s, 18.95 min, and 3 min per epoch respectively. The full SPT pipeline (200 epochs SPT + 100 epochs finetuning) is approximately $3\times$ the cost of from-scratch training for 100 epochs. PathX was excluded because its full pipeline is roughly $16\times$ more expensive than Pathfinder (Section 2). The full research project, including preliminary runs, additional seeds, and configurations not reported, required substantially more compute than the experiments listed in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. We use only publicly available benchmark datasets (LRA, UCR) and a synthetic dataset we generate ourselves (Appendix D); no human subjects, no personal data, and no scraped content are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is foundational research on the training dynamics of Transformers for sequence classification. A better understanding of when label supervision fails to learn Attention patterns may modestly improve sample efficiency of small Transformer models, with corresponding small reductions in compute. We see no direct path to harmful applications beyond those already associated with Transformer training in general; the work does not propose a new application, model release, or dataset.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pre-trained models, generative models, or scraped datasets that would pose a risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original LRA benchmark [Tay et al., 2020], the codebase of Amos et al. [2024], and the source datasets used through LRA: CIFAR-10 [Krizhevsky et al., 2009], IMDb [Maas et al., 2011], ListOps [Nangia and Bowman, 2018], Pathfinder [Linsley et al., 2018], and the ACL Anthology Network [Radev et al., 2013]. For Appendix B.1 we cite the UCR archive via Dempster et al. [2019]. All datasets used are publicly released by their authors under their respective licenses; we use them under the terms provided by the original distributors and follow the usage convention established by Amos et al. [2024].

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or models. The synthetic task is fully specified by the generative procedure in Appendix D and can be reproduced from that description.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as a component of the core methodology, experimental pipeline, or scientific analysis. Any use was limited to incidental writing assistance (grammar and phrasing).

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.