Synthetic Pre-Pre-Training Improves Language Model Robustness to Noisy Pre-Training Data

Xu Guo[1,2,3] **Runyu Peng**[4] **Jian Tong**[3] **Yunhua Zhou**[3]
Haijun Lv[3] **Zhihui Lu**[†1] **Qipeng Guo**[†2,3]

[1]Fudan University [2]Shanghai Innovation Institute
[3]Shanghai AI Laboratory [4]New York University

{pengrunyu, tongjian, zhouyunhua, lvhaijun, guoqipeng}@pjlab.org.cn
guox24@m.fudan.edu.cn lzh@fudan.edu.cn

[†]Corresponding authors

Abstract

Large language models (LLMs) rely on web-scale corpora for pre-training. The noise inherent in these datasets tends to obscure meaningful patterns and ultimately degrade model performance. Data curation mitigates but cannot eliminate such noise, so pre-training corpora remain noisy in practice. We therefore study whether a lightweight pre-pre-training (PPT) stage based on synthetic data with learnable temporal structure helps resist noisy data during the pre-training (PT) stage. Across various corruption settings, our method consistently improves robustness to noise during PT, with larger relative gains at higher noise levels. For a 1B-parameter model, a 65M-token RNN-PPT stage reaches the baseline's final loss with 42.3–51.4% fewer natural-text PT tokens across four noise levels and three seeds. Mechanistic analyses suggest PPT does not immediately suppress attention to noisy tokens. Rather, PPT-initialized models gradually downweight attention between corrupted tokens during noisy PT. This indicates that synthetic PPT inhibits noise self-modeling and shapes the subsequent optimization trajectory. Code is available at `https://github.com/guox18/formal-language-prepretraining.git`.

1 Introduction

Large language models (LLMs) rely on web-scale corpora for pre-training [Penedo et al., 2023, Soldaini et al., 2024, Penedo et al., 2024, Li et al., 2025, Su et al., 2025]. Yet such data inevitably contains noise, from duplicated boilerplate [Elazar et al., 2023] to formatting errors [Zhou et al., 2024]. Investigations show that the noise inherent in these datasets can reduce the model's knowledge capacity, sometimes by up to $20\times$ [Allen-Zhu and Li, 2023, 2024]. More recent studies further show that such noise can destabilize training, obscure predictive structure and degrade downstream performance [Ru et al., 2025, Zhang et al., 2026].

To mitigate these issues, large-scale pre-training pipelines rely heavily on data curation [Raffel et al., 2020, Lee et al., 2022]. However, such curation is inherently incomplete. For instance, automated filters could conflate useful rare documents with low-quality text [Longpre et al., 2023]. This creates an inherent trade-off between filtering out noise and preserving tail knowledge. Furthermore, exhaustive cleaning is computationally prohibitive at web scale [Joulin et al., 2016, Albalak et al., 2024]. This means that pre-training corpora inevitably retain some residual noise. A complementary strategy is to enhance the model's intrinsic robustness to noise during training. While robust learning

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

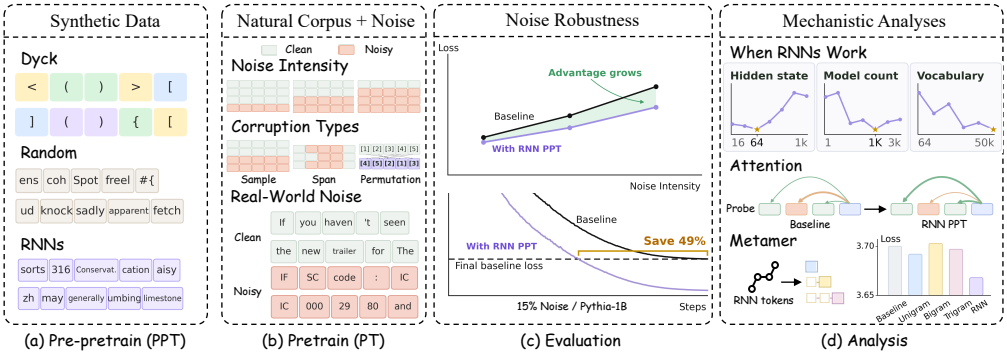

Figure 1: Overview of our setup and main findings. (a) We first run a lightweight PPT stage, drawing sequences from one of three synthetic sequences: Dyck, Random, or randomly initialized RNNs. (b) The model is then pre-trained on natural text under controlled noise, varying both intensity and corruption type, and further evaluated on real-world noise. (c) Top: final loss versus noise intensity. Bottom: training loss over steps. (d) We study when RNN-based PPT works and probe its mechanism through attention analyses and metamer controls. These controls match the unigram, bigram, or trigram statistics of RNN-generated tokens.

from noisy labels is well-established in fields like computer vision [Song et al., 2022], improving the noise tolerance of LLMs during pre-training remains largely underexplored [Ru et al., 2025].

Studies show that pre-trained models are more robust to noisy labels than their from-scratch counterparts [Hendrycks et al., 2019]. This suggests that randomly initialized networks are vulnerable to noise. However, this paradigm relies on pre-training to mitigate noise during downstream fine-tuning, and therefore cannot be applied to our scenario. Fortunately, recent work shows that a brief initial training phase on non-natural data (e.g., formal languages) can effectively instill a structural prior in LLMs [Papadimitriou and Jurafsky, 2023]. However, such non-natural data is typically narrow and homogeneous [Shinnick et al., 2025a]; this contrasts with natural pre-training data, where both meaningful patterns and noise arise from heterogeneous sources.

Building on this insight, we explore whether a pre-pre-training (PPT) stage can enhance robustness to pre-training (PT) noise. Specifically, we introduce a lightweight, synthetic PPT task where the model learns from sequences generated by an ensemble of randomly initialized recurrent neural networks (RNNs). Even without training, RNNs inherently produce sequences with temporal structure [Siegelmann and Sontag, 1992, Jaeger, 2001, Merrill et al., 2020]. Aggregating outputs from many such independent generators exposes the model to a diverse set of structural patterns. This establishes a broad prior for sequential dependencies, guiding the model to prioritize structured signals over unstructured noise during subsequent PT.

This approach is guided by two design principles, which we ablate in Section 5.2. First, the source should be *learnable* for the downstream model, achieved here by using RNNs with moderate hidden state sizes. Second, the source should be *less biased*, motivating our use of a broad ensemble and the full vocabulary to generate such RNN sequences. Figure 1 summarizes our setup and main findings.

As noise-robust pre-training for LLMs remains underexplored and lacks established baselines, we compare our method against two representative alternatives: an unstructured *Random* i.i.d. token source and a formal-language *Dyck* source. Across these comparisons, our method consistently outperforms both alternatives, substantially improving model robustness to various types of noise, ranging from controlled corruptions to naturally noisy web data.

The contributions of this paper are threefold:

- **A synthetic PPT recipe for noise-robust pre-training.** We show that a short PPT stage on random RNN sequences markedly improves the model's noise tolerance during pre-training. The relative gains grow with noise intensity, and transfer across diverse corruption types and naturally noisy web corpora. Across three seeds and the tested noise levels, observed per-seed PT-token savings range from 8.3–21.3% at 160M parameters to 42.3–51.4% at 1B parameters.

- **Design choices that make our RNN-PPT effective.** Through controlled ablations, we identify three conditions under which RNN-PPT yields gains: generators should remain learnable, the ensemble should be large enough to avoid idiosyncratic bias, and the vocabulary should be broad. Together, these conditions reduce RNN-PPT design to a small set of actionable design choices.
- **Mechanistic characterization of the robustness gain.** We find that the gain emerges during noisy PT, not at PPT initialization. An attention probe shows that RNN-PPT models progressively learn to downweight attention between corrupted tokens—an effect concentrated in late-layer heads. In the source-control experiments, the original RNN generator yields larger validation-loss gains than the tested n-gram metamers, simple linear-state sources, nonlinear RNNs, Dyck, and randomly initialized Transformers.

2 Related Work

2.1 Noise in Pre-Training

Web-scale pre-training corpora, spanning domains such as healthcare, law, and financial technology, are unavoidably noisy [Elazar et al., 2023, Zhou et al., 2024], which reduces knowledge capacity and degrades downstream performance [Allen-Zhu and Li, 2023, 2024, Zhang et al., 2026]. Curation pipelines [Raffel et al., 2020, Wenzek et al., 2019, Rae et al., 2022, Penedo et al., 2023, Lee et al., 2022, Gunasekar et al., 2023] mitigate but cannot eliminate residual noise: automated filters risk discarding tail knowledge [Longpre et al., 2023], and exhaustive cleaning is prohibitive at web scale [Joulin et al., 2016, Albalak et al., 2024]. While Noisy Model Learning has been studied extensively in supervised image classification [Song et al., 2022], the intrinsic noise tolerance of LLMs during pre-training remains underexplored [Ru et al., 2025]; this paper addresses that gap.

2.2 Pre-Training on Non-Linguistic Data

Pre-training on non-linguistic data injects structural inductive biases into language models before they see natural text [Papadimitriou and Jurafsky, 2023]. This direction remains largely underexplored, yet the few existing studies offer illuminating insights. Papadimitriou and Jurafsky [2020, 2023] first show that pre-training on non-linguistic data such as music or code can transfer structural priors to natural language. Subsequent work extends this idea to algorithmic data. Hu et al. [2025] study Dyck-language variants that accelerate natural language modeling, and explain the effect through the Chomsky hierarchy [Delétang et al., 2023]. Shinnick et al. [2025b], Jiang et al. [2026] suggest that procedural pre-training induces localized, composable structures across attention and MLP components. Bloem [2025] examines how pre-training aids multi-task learning, and explains its effects through Solomonoff induction. In this paper, we follow this paradigm but shift the focus to exploring its potential for enhancing noise robustness during pre-training.

3 Method

3.1 Two-Stage Training Pipeline

Our training pipeline consists of two stages: a short pre-pre-training (PPT) phase on synthetic data for S_{ppt} steps, followed by standard pre-training (PT) on natural text for S_{pt} steps. We evaluate models based on their final validation loss on the PT corpus. To isolate the impact of PPT on noise robustness, we keep the PT data and training recipe fixed across all methods, varying only the synthetic data used during PPT and the resulting parameter initialization.

3.2 RNN Synthetic PPT Source

Our proposed source samples sequences from an ensemble of randomly initialized RNN models. Because each generator is a fixed recurrent system, the resulting sequences remain noise-like on the surface while containing temporal structure. Training on sequences from multiple such generators exposes the model to diverse structural patterns before it encounters noisy natural text. Concretely, for each synthetic sequence we choose a generator $g \sim \mathrm{Unif}(\{1, \ldots, M\})$ and initialize $x_0 \sim \mathrm{Unif}(\{1, \ldots, V\})$, $h_0^{(g)} = \mathbf{0}$, where V is the tokenizer vocabulary size. We then run the fixed

recurrence $h_t^{(g)} = A^{(g)} e_{x_{t-1}} + W^{(g)} h_{t-1}^{(g)} + b^{(g)}$ with logits $\ell_t^{(g)} = C^{(g)} h_t^{(g)} + d^{(g)}$, where $e_{x_{t-1}} \in \{0,1\}^V$ is the one-hot encoding of the previous token. The next token is then sampled as

$$x_t \sim \text{Categorical}\Big(\text{softmax}\Big(\ell_t^{(g)}/\tau\Big)\Big), \tag{1}$$

where τ is the sampling temperature. Unless ablated, we use M=1000 generators, hidden size H=64, the full vocabulary (V=50,304), and τ=1. All generator parameters are sampled once at random and then kept fixed; Appendix I gives the initialization details. We explore the impact of specific design choices (such as the number of generators) in our ablation studies (Section 5).

3.3 Noise Injection

Following prior studies on noisy pre-training [Ru et al., 2025, Zhang et al., 2026], we apply controlled corruption to a curated corpus, using C4 as a relatively clean baseline. Our primary setting introduces sample-level corruption during PT: with probability p, each training sequence is replaced by tokens sampled uniformly from the vocabulary, preserving the original batch size, sequence length, and optimizer settings (full protocol in Appendix A).

While sample-level corruption is a controlled proxy for real web noise, we broaden our evaluation with two finer-grained variants—token permutation and span corruption, which perturb token segments rather than entire sequences (Figure 1(b)). To study real web noise, we follow standard practices [Ankner et al., 2024] by using an external reference model to partition FineWeb into clean and noisy subsets (details in Appendix K).

4 Experiments

4.1 Experimental Setup

Datasets. Our primary pre-training corpus is C4 [Raffel et al., 2020]. To evaluate robustness on real web noise, we also use FineWeb [Penedo et al., 2024] as a natural language dataset.

Baselines. We compare four PPT settings: no PPT (pre-training from scratch), Random PPT, Dyck PPT, and our RNN PPT. Random PPT serves as a structure-free comparison, sampling tokens independently from the full vocabulary for the same PPT steps. Dyck PPT uses k-Shuffle Dyck, the strongest formal-language source from prior work [Hu et al., 2025], which interleaves k independent, balanced 1-Dyck bracket-matching strings. Following prior work, we set k=64.

Models. In our main experiments, we pre-train a 160M-parameter Pythia model [Biderman et al., 2023], which enables multi-seed sweeps over noise levels and ablations. To test whether the effect persists at a larger model size, we also train a 1B-parameter model. Appendix J reports the runtime costs for both scales.

Training. Unless stated otherwise, we run $S_{\text{ppt}} = 500$ steps of PPT before standard PT. Architecture and hyperparameters remain fixed across comparisons. For token-efficiency comparisons, all experiments use packing, so equal steps correspond to equal training tokens.

Evaluation. Our main metric is held-out validation loss on the clean PT corpus, following standard practice in studies of pre-training dynamics [Kaplan et al., 2020] and noisy pre-training data [Zhang et al., 2026]. We also report *PT-token savings*: the reduction in PT tokens required to match the baseline's final loss. As supplementary downstream evidence, we evaluate the final base checkpoints on LAMBADA-OpenAI [Paperno et al., 2016], WikiText-2 [Merity et al., 2016], and zero-shot PIQA [Bisk et al., 2019], HellaSwag [Zellers et al., 2019], ARC-Easy/Challenge [Clark et al., 2018], and WinoGrande [Sakaguchi et al., 2020]. Results use three matched seeds; Appendix B gives the protocols and task-level results. See Appendix A for full experimental details.

4.2 Main Results

Figure 2 shows the controlled-noise results on C4 at 160M scale. RNN-PPT consistently achieves a lower final validation loss than the baseline, with the gap widening at higher noise levels. Across-seed variation is small throughout; for example, in the clean setting, the final-loss standard deviations

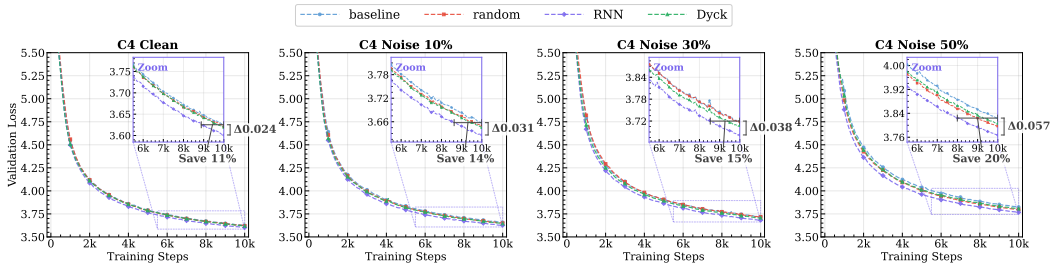

Figure 2: Main controlled-noise results on C4, averaged over three seeds. RNN-PPT consistently reaches a lower validation loss, and its advantage grows with the corruption level. Annotations report the final-loss gap and the PT-token savings needed for RNN-PPT to match the baseline. Dyck-PPT shows the same trend but more weakly, and Random PPT closely tracks the baseline.

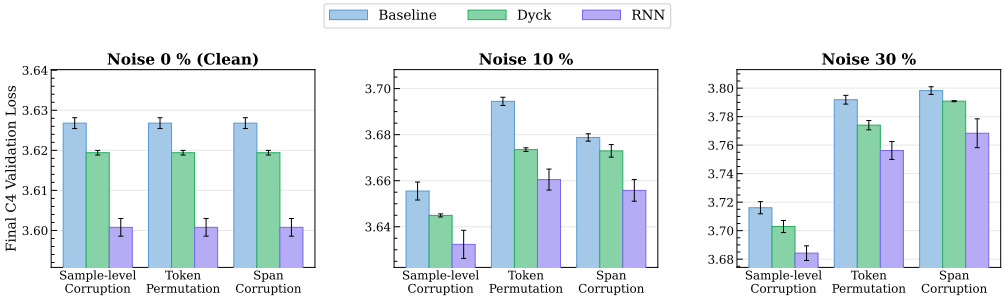

Figure 3: Generalization across corruption types. RNN-PPT improves final loss under all three protocols (sample-level corruption, token permutation, and span corruption), showing that the benefit is not tied to any single corruption type. Bars show mean final validation loss over three seeds; error bars denote standard deviation.

are 0.001 for the baseline and 0.002 for RNN-PPT (Appendix Table 10). The observed PT-token savings span 8.3–21.3% across the tested noise levels and three seeds (Appendix Table 3). Dyck-PPT offers modest improvements, while Random PPT performs similarly to the baseline, indicating that structure, not just warm-up, is necessary for robustness.

4.3 Generalization Across Noise Types

We extend our evaluation beyond sample-level corruption to token permutation and span corruption (Figure 3). RNN-PPT consistently improves final validation loss across all three noise types, with the most significant gains under token permutation. Dyck-PPT provides smaller benefits. This confirms that RNN-PPT's robustness is agnostic to the specific corruption protocol.

4.4 Robustness on FineWeb

We next test whether the robustness gain transfers beyond C4, first under a controlled FineWeb corruption protocol and then on naturally noisy FineWeb subsets.

Controlled Noise. We first apply the same controlled-noise protocol to FineWeb (Figure 4). RNN-PPT achieves the lowest final validation loss at every tested noise level, outperforming both the baseline and Dyck-PPT. Thus, the effect is not specific to C4.

Naturally Noisy Data. We then partition FineWeb into clean and noisy subsets using an external reference model [Ankner et al., 2024] (Appendix K). The noisy subset contains naturally occurring, less informative web fragments; representative cases appear in Appendix L. RNN-PPT lowers validation loss on both splits (Figure 5), with the larger gain on the noisier split. Dyck-PPT has little effect in this setting, suggesting that RNN-PPT transfers more effectively to natural web noise.

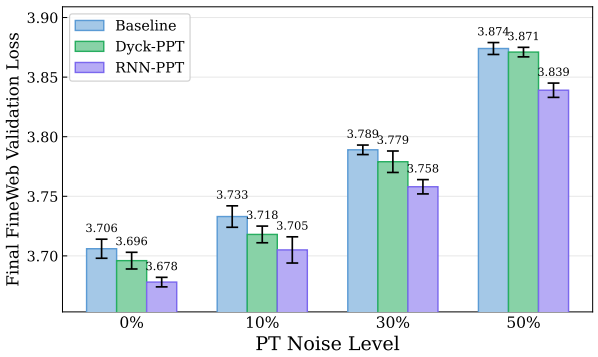

Figure 4: Controlled-noise FineWeb results. Bars show mean final FineWeb validation loss over three seeds and error bars denote standard deviation. RNN-PPT is best at every tested PT noise level.

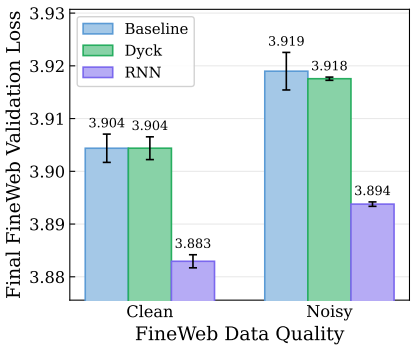

Figure 5: Validation on naturally noisy data. RNN-PPT lowers validation loss on both quality splits, with a larger gain on the noisier one.

4.5 Robustness Persists at Larger Scale

Figure 8 tests whether the effect persists at 1B-parameter scale. Since the smaller 160M experiments already cover a broad range of corruption levels, the 1B experiment focuses on a finer-grained and more practical noise range. RNN-PPT consistently outperforms the baseline at every tested noise level, despite the larger model capacity and longer PT duration. Appendix D shows the same trend when extending the 160M PT budget from 10K to 20K steps.

We also include a budget-matched C4-PPT control to separate the effect of the RNN source from the effect of seeing additional clean, PT-like data. C4-PPT yields only marginal gains at 1B. Appendix C reports the corresponding 160M comparison, where RNN-PPT remains the strongest method. This suggests that the robustness gain is not explained by extra C4 training.

The magnitude of the 1B gain is practically meaningful and consistent across noise. At the 1B scale, RNN-PPT reduces final validation loss by 0.10–0.15 across the tested noise range. After the 65M-token RNN-PPT stage, the model reaches the baseline's final loss with 42.3–51.4% fewer natural-text PT tokens across all tested noise levels and seeds. The mean savings are 44.1%, 45.5%, 47.2%, and 48.1% at 0%, 5%, 10%, and 15% noise, respectively (Appendix Table 3). On the final 1B base checkpoints, RNN-PPT lowers WikiText-2 word perplexity by 13.0–17.3% and raises the mean zero-shot accuracy across five tasks by 0.88–1.05 percentage points over the four PT noise levels. The clearest accuracy gains are on PIQA and HellaSwag; ARC-Challenge and WinoGrande show smaller or mixed changes (Appendix B). Two further architecture checks find positive mean final-loss reductions with RNN-PPT for randomly initialized Qwen2.5 and Gated DeltaNet models under both clean and 30%-corrupted PT (Appendix E).

5 Ablation Study

In this section, we analyze which parts of the RNN-PPT design contribute to its robustness: the PPT budget (§5.1) and RNN generator choices (§5.2).

5.1 PPT Budget

We vary the synthetic warm-up length in Figure 6. The benefit emerges within a few hundred steps and then plateaus. At 30% noise, RNN-PPT is slightly worse than the baseline at 100 steps, matches it around 200 steps, and improves further to peak near 500 steps; longer PPT brings no further gain. Dyck-PPT exhibits a similar but less pronounced trend. We therefore adopt 500 PPT steps as the default PPT budget: a lightweight intervention that already captures most of the robustness gain.

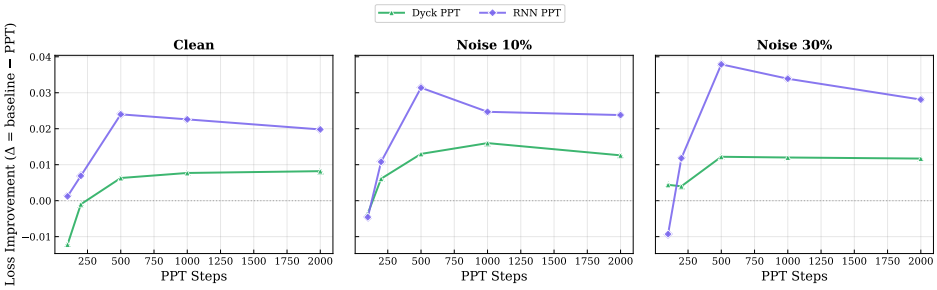

Figure 6: Sensitivity to PPT budget. Improvements over the baseline appear after a few hundred synthetic steps and then largely plateau. For RNN-PPT, 500 steps is close to the best PPT budget across noise levels.

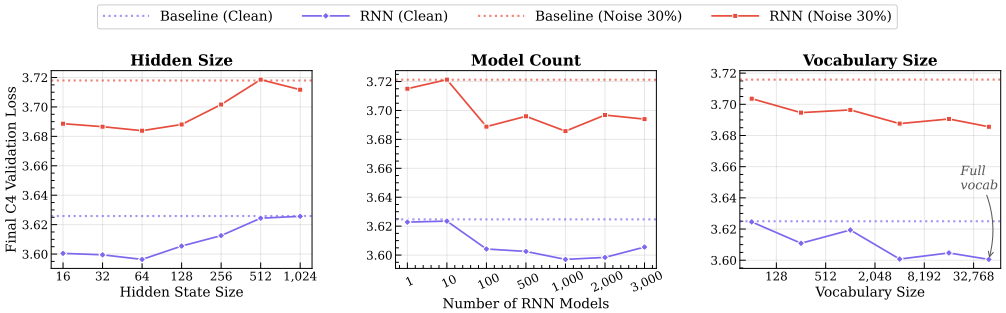

Figure 7: RNN design ablations at 0% and 30% PT noise. **Left:** Transfer is strongest within a moderate generator-complexity range. **Middle** and **Right:** Larger ensembles and broader vocabularies yield the best overall robustness, supporting our default of a large ensemble and full vocabulary.

5.2 RNN Generator Design

Section 1 motivated RNN-PPT with two design principles: the synthetic source should be learnable by the downstream model, and it should be less biased. We test learnability by varying RNN hidden size, and test low-bias design by varying ensemble size and vocabulary size. Figure 7 shows the 0% and 30% PT settings for readability; Appendix G reports the complementary 10% and 50% results.

Learnability. The hidden-size sweep (Figure 7, Left) shows a broad sweet spot. Hidden size 64 attains the lowest final loss under 30% noise and remains near the best at 10% noise, while the 16–64 range is strongest at 50%. With much larger generators (512/1024 hidden sizes), the gain over the baseline nearly disappears. Thus, RNN-PPT helps when the generated sequences are relatively simple for the downstream model to learn, but not when the generator becomes too complex.

Low-bias source design. The generator-count and vocabulary sweeps support the low-bias principle from two angles. For generator count (Figure 7, Middle), one or ten generators yield little benefit; gains emerge around 100 generators and remain effective through 3000. Appendix G shows the same trend at 10% and 50% noise, with 1000 generators performing best. This suggests that too few generators might transfer idiosyncratic recurrent patterns, while a larger ensemble provides a broader sequential prior.

The vocabulary sweep (Figure 7, Right) shows a similar pattern for token support. Restricted vocabularies recover only part of the benefit, whereas larger vocabularies generally work better. Since the full vocabulary performs best in the clean setting and in most noisy settings, we use it as the default. Together, these results suggest that RNN-PPT is most effective when its structure is broad rather than tied to a small generator set or a narrow token subset.

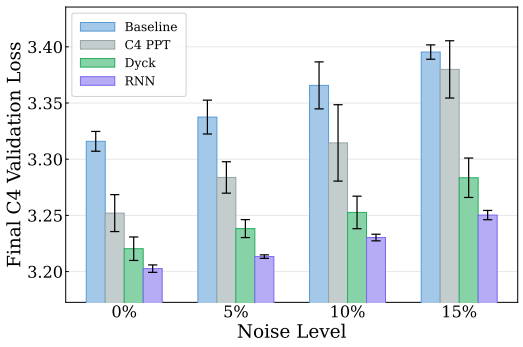

Figure 8: Method comparison at 1B scale. With a 25K-step PT budget, RNN-PPT remains effective at larger model size across all tested noise levels.

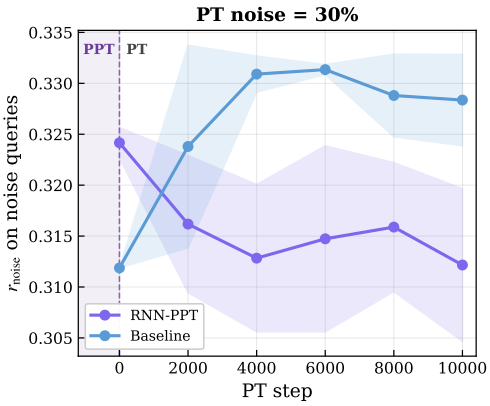

Figure 9: Layer-mean noise self-modeling, measured as r_{noise} on noisy queries over PT steps at 30% noise. Shaded band: $\pm$sd across three seeds.

6 Analysis

The preceding experiments demonstrate the effectiveness and robustness of RNN-PPT under varied conditions. We now use attention probes to examine whether models rely on corrupted tokens to predict other corrupted tokens, and source controls to assess which properties of synthetic sequences accompany the transfer gain.

6.1 Noise Self-Modeling in Attention

Models trained from scratch are more vulnerable to noisy data. We hypothesize that this vulnerability arises because they learn to model the noise itself as a predictable pattern, relying on earlier corrupted tokens to process subsequent ones. We refer to this behavior as *noise self-modeling*.

Specifically, we investigate whether models use noisy tokens to predict other noisy tokens under noisy PT. We ask (i) *when* the gap between RNN-PPT and the baseline emerges during PT, and (ii) *where* in the model this difference is localized. The probe setup is given in Appendix H.

Metric. We define an attention-based metric to quantify noise self-modeling, which captures the extent to which representations at noisy positions are constructed from preceding corrupted positions in the causal context.

Concretely, let a_{qk} denote the attention weight from query position q to key position k. For a given q, let $K_{\text{valid}}(q)$ be all valid key positions in its causal prefix, and let $K_{\text{noise}}(q) \subseteq K_{\text{valid}}(q)$ be the noisy ones. Since attention weights are normalized over the valid prefix, $\sum_{k \in K_{\text{valid}}(q)} a_{qk} = 1$, the inner sum in Equation 2 is the fraction of query q's attention mass assigned to noisy keys. We average this quantity over the noisy query set Q_{noise}:

$$r_{\text{noise}} = \frac{1}{|Q_{\text{noise}}|} \sum_{q \in Q_{\text{noise}}} \sum_{k \in K_{\text{noise}}(q)} a_{qk} \,. \tag{2}$$

We define $\Delta r_{\text{noise}} = r_{\text{noise}}^{\text{RNN-PPT}} - r_{\text{noise}}^{\text{No-PPT}}$ per (layer, head) and average across three model seeds.

Figure 9 tracks the layer-mean r_{noise} on noisy queries across PT checkpoints. No-PPT models begin from the randomly initialized checkpoint. As noisy PT proceeds, the no-PPT models increasingly attend from noisy query tokens to noisy key tokens. This suggests that the model learns to treat corrupted tokens as predictive of other corrupted tokens. RNN-PPT changes this trajectory: within the first 2K PT steps, its curve drops below the no-PPT curve, and the gap remains throughout PT. This indicates that RNN-PPT suppresses noise self-modeling during PT. The reduction in noise self-modeling emerges early and persists, rather than appearing as a late-training correction.

Figure 10 visualizes Δr_{noise} per (layer, head) for checkpoints trained with PT noise, using a fixed probe setting. The effect is not spread evenly across heads: the largest reductions in noise self-

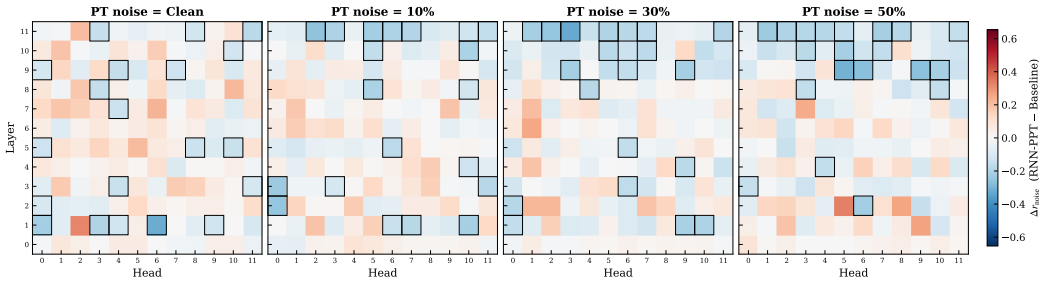

Figure 10: Seed-mean per-(layer, head) Δr_{noise} on noisy query tokens under the fixed probe setting described in Appendix H. Panels vary the PT noise level from clean to 50%. Blue indicates weaker noise self-modeling for RNN-PPT than for models without PPT at that head. Black borders mark the top-20 most-negative heads per panel.

modeling concentrate in layers 8–11, and intensify as PT noise rises. The effect is localized rather than uniform, indicating that later layers drive the reduction in noise self-modeling. This suggests PPT mitigates noise during high-level integration rather than at shallow processing stages.

Additional probes and attention interventions. Across five matched seeds of the 160M models after 10K PT steps at 30% noise, RNN-PPT reduces noise-to-noise attention in layers 9–11 under all five tested corruption probes. Scaling this attention route down and renormalizing the attention rows lowers clean-token loss, while scaling it up raises loss, in every tested seed and model (No-PPT, Dyck-PPT, and RNN-PPT). The effect concentrates on clean targets immediately following corrupted tokens. These interventions support a role for this route in recovery from corrupted context; they do not establish how much of the PPT training gain it explains (Appendix H.1).

6.2 Unigram, Bigram, and Trigram Metamer Controls

A natural concern is whether RNN-PPT's benefit comes mainly from low-order token statistics rather than from richer sequential structure. To test this possibility, we construct unigram, bigram, and trigram metamers from the RNN source subset for 500 PPT steps. For each order, we fit the token-level model to the RNN source tokens and then sample new sequences from it, preserving local statistics while removing the original long-range organization. We then re-run the full pipeline with each metamer as the PPT source, holding the PPT budget and PT recipe fixed. Table 1 reports the mean improvement over the no-PPT baseline across three seeds; construction details are given in Appendix M. These metamers produce only marginal gains. Simple linear-state sources, nonlinear RNNs, and a randomly initialized Transformer also yield smaller gains (Appendix F). Together, these controls indicate that the gain is sensitive to the synthetic source design: preserving local n-gram statistics, adding simple recurrence, or increasing generator complexity alone does not reproduce the original RNN-PPT gain under the tested PPT budget.

PPT source	0%	10%	30%	50%
Unigram metamer	0.006	0.006	0.008	0.014
Bigram metamer	−0.002	0.002	−0.011	−0.001
Trigram metamer	0.002	0.003	0.004	0.005
RNN subset	**0.026**	**0.029**	**0.036**	**0.052**

Table 1: Unigram, bigram, and trigram metamer diagnostic. Entries report mean improvement over the no-PPT baseline in final validation loss across three seeds.

7 Discussion

How does RNN-PPT impart such robustness against noisy data? A natural explanation is that learning synthetic sequential structure during PPT makes the model downweight noise. Our attention analyses in Section 6 do not support this. At the start of PT, RNN-PPT models attend more strongly among corrupted tokens than models without PPT, suggesting that PPT has not taught the model to directly downweight noise. The benefit emerges during noisy PT: RNN-PPT models quickly reduce noise-to-noise attention, while models without PPT drift in the opposite direction. Thus, RNN-PPT

improves robustness not by directly downweighting noisy tokens during PPT, but by guiding the subsequent optimization trajectory toward lower noise self-modeling.

More broadly, PPT acts as a complement rather than a replacement for data curation. Curation intervenes on the corpus, but remains incomplete in practice and can leave residual noise. PPT intervenes on the model: it leaves the corpus and PT recipe untouched, yet makes training more tolerant of noisy mixtures.

In this work, motivated by the hypothesis in Section 1, we chose randomly initialized RNNs as the synthetic source and validated this choice empirically. We note three limitations that point to natural follow-ups. First, testing each candidate source requires running the full PPT-to-PT pipeline, making source selection costly; a useful future direction is to identify statistics that predict downstream efficacy without running the full pipeline. Second, our experiments stop at 1B parameters and 25K PT steps under compute constraints (Appendix J reports wall-clock costs); establishing scaling laws for PPT-induced robustness is a natural follow-up. Third, our source controls cover only a small set of generators and do not isolate the sequence statistics responsible for transfer; testing broader source families remains a next step.

8 Conclusion

We demonstrate that synthetic PPT improves robustness to noisy PT data without modifying the PT recipe. We evaluate this effect under controlled corruption, multiple noise types, real web noise, and model scales up to 1B parameters. Across these settings, a lightweight RNN-based PPT stage consistently yields the strongest gains, with benefits that grow as pre-training becomes noisier.

Our experiments point to two design principles. First, the synthetic source should be learnable but not trivial: overly complex generators transfer poorly, while preserving local unigram-to-trigram statistics is insufficient to reproduce the full RNN-PPT gain. Second, the source should be less biased: increasing the number of RNN generators and broadening the vocabulary both improve robustness. Our attention analysis offers one explanation. PPT does not reduce the model's attention to noise before PT; rather, RNN-PPT initially increases attention among corrupted tokens. Its advantage appears during noisy PT: RNN-PPT guides training away from noise self-modeling, whereas models trained without PPT increasingly model the noise itself.

Acknowledgments and Disclosure of Funding

The authors thank the Area Chairs, Program Chairs, and anonymous reviewers for their constructive comments. The authors also thank Shanghai Innovation Institute for its support. This work was supported by Shanghai AI Laboratory, in part by the Yangtze River Delta Science and Technology Innovation Community Joint Research Project under Grant YDZX20233100004031, and in part by the National Natural Science Foundation of China under Grants 72595845 and 72595840. The authors declare no competing interests.

References

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L. Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models, 2024. URL `https://arxiv.org/abs/2405.20541`.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya

Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Peter Bloem. Universal pre-training by iterated random computation, 2025. URL https://arxiv.org/abs/2506.20057.

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

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the chomsky hierarchy, 2023. URL https://arxiv.org/abs/2207.02098.

Yanai Elazar, Akshita Bhagia, Ian Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, et al. What's in my big data? *arXiv preprint arXiv:2310.20707*, 2023.

Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, May 2023. URL https://github.com/openlm-research/open_llama.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023. URL https://arxiv.org/abs/2306.11644.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty, 2019. URL https://arxiv.org/abs/1901.09960.

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases, 2025. URL https://arxiv.org/abs/2502.19249.

Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German national research center for information technology gmd technical report*, 148(34):13, 2001.

Liangze Jiang, Zachary Shinnick, Anton van den Hengel, Hemanth Saratchandran, and Damien Teney. Procedural pretraining: Warming up language models with abstract data, 2026. URL https://arxiv.org/abs/2601.21725.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better, 2022. URL https://arxiv.org/abs/2107.06499.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao

Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. DataComp-LM: In search of the next generation of training sets for language models, 2025. URL https://arxiv.org/abs/2406.11794.

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality and toxicity, 2023. URL https://arxiv.org/abs/2305.13169.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A Smith, and Eran Yahav. A formal hierarchy of rnn architectures. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 443–459, 2020.

Isabel Papadimitriou and Dan Jurafsky. Learning music helps you read: Using transfer to study linguistic structure in language models, 2020. URL https://arxiv.org/abs/2004.14601.

Isabel Papadimitriou and Dan Jurafsky. Injecting structural hints: Using language models to study inductive biases in language learning, 2023. URL https://arxiv.org/abs/2304.13060.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023. URL https://arxiv.org/abs/2306.01116.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2022. URL https://arxiv.org/abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Jinghan Ru, Yuxin Xie, Xianwei Zhuang, Yuguo Yin, Zhihui Guo, Zhiming Liu, Qianli Ren, and Yuexian Zou. Do we really have to filter out random noise in pre-training data for language models?, 2025. URL https://arxiv.org/abs/2502.06604.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8732–8740, 2020. URL https://aaai.org/papers/08732-winogrande-an-adversarial-winograd-schema-challenge-at-scale/.

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

Zachary Shinnick, Liangze Jiang, Hemanth Saratchandran, Anton van den Hengel, and Damien Teney. Transformers pretrained on procedural data contain modular structures for algorithmic reasoning, 2025b. URL https://arxiv.org/abs/2505.22308.

Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449, 1992.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE transactions on neural networks and learning systems*, 34(11):8135–8153, 2022.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset, 2025. URL https://arxiv.org/abs/2412.02595.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data, 2019. URL https://arxiv.org/abs/1911.00359.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.06464.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019. URL https://aclanthology.org/P19-1472/.

Qizhen Zhang, Ankush Garg, Jakob Foerster, Niladri Chatterji, Kshitiz Malik, and Mike Lewis. An empirical study on noisy data and llm pretraining loss divergence, 2026. URL https://arxiv.org/abs/2602.02400.

Jing Zhou, Chenglin Jiang, Wei Shen, Xiao Zhou, and Xiaonan He. Leveraging web-crawled data for high-quality fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11297–11312, 2024.

Setting	Main	Scale-up
Model size	160M	1B
PPT steps	500	1000
PT steps	10k	25k
Batch size	16	16
Gradient accumulation	2	2
Effective batch size	32	32
Sequence length	2048	2048
Learning rate	7×10^{-4}	5×10^{-4}
LR schedule	Cosine w/ warmup	Cosine w/ warmup
Warmup steps	500	500
Weight decay	0.1	0.1
Gradient clipping	1.0	1.0
Optimizer	AdamW	AdamW
β_1, β_2	0.9, 0.999	0.9, 0.999
ϵ	10^{-6}	10^{-6}
Mixed precision	bf16	bf16

Table 2: Summary of the main training configurations.

A Additional Experimental Details

Table 2 summarizes the main experimental configurations used throughout the paper.

We use 1000 PPT steps for these 1B runs, based on a small pilot sweep at this scale; gains beyond 1000 were marginal. All experiments reported here were run with Hugging Face Transformers 5.5.0 and Datasets 4.8.4.

Unless explicitly noted otherwise, both PPT and PT runs use `packing_mode=pack`. Therefore, the number of tokens used for the same number of steps is effectively the same.

Controlled corruption variants. For token permutation, each 2048-token sequence is partitioned into non-overlapping windows of size 32. We randomly select windows until the number of covered tokens reaches the target corruption rate, and independently permute tokens within each selected window. For span corruption, we repeatedly sample span lengths uniformly from integers 5–20 and random start positions until the union of selected span positions reaches the target corruption rate. Tokens in the selected positions are replaced with tokens sampled uniformly from the vocabulary.

Block training and LR schedule. We adopt a two-stage block-training protocol in which PPT and PT are two independent optimization runs, each with its own cosine-with-warmup schedule and its own 500-step linear warmup (see Table 2).

Our PPT stage is lightweight: at 1B scale it occupies $S_{\text{ppt}} = 1000$ steps against a PT budget of $S_{\text{base}} = 25\text{K}$ steps, i.e. 4% of the baseline's PT budget (and 5% at 160M, where $S_{\text{ppt}} = 500$ and $S_{\text{base}} = 10\text{K}$). We report PT-token savings, excluding PPT tokens. If a PPT method first reaches the baseline's final validation loss after S_{match} PT steps, estimated by linear interpolation between adjacent evaluations, the reported saving is $1 - S_{\text{match}}/S_{\text{base}}$. Under our packed, equal-token-per-step setup, subtracting the PPT cost from the saved PT tokens would lower the reported percentages by 4 points at 1B and 5 points at 160M.

B Downstream Language Evaluation

Table 4 and Table 5 report LAMBADA perplexity for the main comparison methods. We focus on perplexity because it provides a more fine-grained language-modeling signal. All reported entries are means and standard deviations over three seeds. Lower is better. For reference, these perplexities are far from random prediction: with a vocabulary of roughly 50k tokens, a uniform-random predictor would have perplexity on the order of 50,000, whereas the values reported here are in the 76–797 range.

Model	PT noise	Mean PT-token saving (%)	Across-seed range (%)
160M	0%	12.0 ± 0.7	11.2–12.4
160M	10%	10.6 ± 2.8	8.3–13.7
160M	30%	13.1 ± 4.1	8.5–16.1
160M	50%	18.0 ± 4.7	12.6–21.3
1B	0%	44.1 ± 1.5	42.4–45.3
1B	5%	45.5 ± 3.0	42.3–48.4
1B	10%	47.2 ± 3.7	45.0–51.4
1B	15%	48.1 ± 1.9	46.1–49.8

Table 3: PT-token savings for RNN-PPT on C4. Values are mean $\pm$ standard deviation and full range over three seeds. Savings exclude the PPT tokens. The 160M and 1B runs use different PT horizons (10K and 25K steps), so their savings should not be interpreted as a scale-controlled comparison.

Method	0%	10%	30%	50%
Baseline	404.8 ± 45.9	404.9 ± 42.6	489.0 ± 32.1	796.7 ± 72.1
Dyck PPT	358.1 ± 40.6	411.2 ± 92.0	511.7 ± 52.7	690.2 ± 130.2
RNN PPT	$\mathbf{337.6 \pm 39.0}$	$\mathbf{361.2 \pm 75.8}$	$\mathbf{457.0 \pm 63.4}$	$\mathbf{589.9 \pm 8.4}$

Table 4: LAMBADA perplexity for the main comparison methods at 160M scale.

To complement LAMBADA with a standard zero-shot downstream benchmark, we evaluate on PIQA [Bisk et al., 2019], a multiple-choice task in which the model scores two candidate completions of a prompt and the higher-likelihood option is selected (random chance is 0.5). Table 6 and Table 7 report PIQA normalized accuracy. We place these results in the appendix for the same reason as LAMBADA: they serve as supplementary evidence rather than the primary basis of our claim. Higher is better.

Additional 1B base-checkpoint evaluations. We evaluate the final 25K-step checkpoints at 0%, 5%, 10%, and 15% PT noise on three matched seeds. WikiText-2 [Merity et al., 2016] word perplexity uses rolling log-likelihood evaluation (Table 8); lower is better. The relative reduction is computed within each seed before averaging. The five-task zero-shot mean in Table 9 equally weights PIQA [Bisk et al., 2019], HellaSwag [Zellers et al., 2019], ARC-Easy, ARC-Challenge [Clark et al., 2018], and WinoGrande [Sakaguchi et al., 2020]. Higher accuracy is better. PIQA, HellaSwag, and ARC use normalized accuracy; WinoGrande uses accuracy. Unless marked as approximate, paired differences and task averages are computed within each seed, then summarized across seeds.

Method	0%	5%	10%	15%
Baseline	113.7 ± 6.4	130.0 ± 3.9	141.8 ± 7.1	159.7 ± 5.1
C4 PPT	97.5 ± 9.9	111.6 ± 9.7	122.4 ± 14.3	159.9 ± 18.6
Dyck PPT	84.6 ± 6.1	91.4 ± 5.2	95.7 ± 8.3	107.4 ± 7.2
RNN PPT	$\mathbf{76.5 \pm 0.6}$	$\mathbf{83.1 \pm 3.4}$	$\mathbf{86.6 \pm 1.6}$	$\mathbf{94.2 \pm 4.1}$

Table 5: LAMBADA perplexity for the main comparison methods at 1B scale.

Method	0%	10%	30%	50%
Baseline	0.605 ± 0.007	0.599 ± 0.005	0.596 ± 0.004	0.585 ± 0.007
Dyck PPT	0.607 ± 0.007	0.600 ± 0.003	0.596 ± 0.005	0.587 ± 0.007
RNN PPT	$\mathbf{0.608 \pm 0.006}$	$\mathbf{0.605 \pm 0.002}$	$\mathbf{0.597 \pm 0.007}$	$\mathbf{0.592 \pm 0.005}$

Table 6: PIQA zero-shot normalized accuracy at 160M scale. Values are mean $\pm$ standard deviation over three seeds. Random chance is 0.5.

Method	0%	5%	10%	15%
Baseline	0.627 ± 0.005	0.626 ± 0.002	0.617 ± 0.011	0.616 ± 0.007
C4 PPT	0.640 ± 0.006	0.631 ± 0.008	0.629 ± 0.007	0.623 ± 0.002
Dyck PPT	0.639 ± 0.008	0.634 ± 0.004	0.632 ± 0.002	0.632 ± 0.009
RNN PPT	$\mathbf{0.642 \pm 0.003}$	$\mathbf{0.641 \pm 0.004}$	$\mathbf{0.636 \pm 0.001}$	$\mathbf{0.632 \pm 0.003}$

Table 7: PIQA zero-shot normalized accuracy at 1B scale. Values are mean $\pm$ standard deviation over three seeds. Random chance is 0.5.

PT noise	No PPT word PPL	RNN-PPT word PPL	Paired relative reduction (%)
0%	55.23 ± 0.82	48.06 ± 0.49	12.98 ± 1.12
5%	57.28 ± 1.20	49.04 ± 0.25	14.36 ± 1.92
10%	58.94 ± 1.48	49.79 ± 0.06	15.48 ± 2.12
15%	61.37 ± 0.75	50.75 ± 0.26	17.29 ± 1.39

Table 8: WikiText-2 word perplexity at 1B scale, reported as mean $\pm$ standard deviation over three matched seeds. RNN-PPT lowers perplexity in every paired seed/noise comparison.

Task	PT noise	No PPT (%)	RNN-PPT (%)	Δ (pp)
PIQA	0%	62.75 ± 0.52	64.2 ± 0.3	≈ 1.45
HellaSwag	0%	30.18 ± 0.26	31.82 ± 0.47	1.64 ± 0.25
ARC-Easy	0%	37.18 ± 0.53	37.79 ± 0.57	0.62 ± 0.32
ARC-Challenge	0%	22.01 ± 1.19	22.67 ± 0.52	0.65 ± 0.94
WinoGrande	0%	50.75 ± 0.80	51.59 ± 0.28	0.84 ± 0.77
Five-task mean	0%	40.57 ± 0.47	≈ 41.61	≈ 1.04
PIQA	5%	62.60 ± 0.19	64.07 ± 0.36	1.47 ± 0.44
HellaSwag	5%	30.08 ± 0.10	31.52 ± 0.06	1.44 ± 0.14
ARC-Easy	5%	36.49 ± 0.57	37.65 ± 0.49	1.16 ± 0.42
ARC-Challenge	5%	21.67 ± 0.52	22.98 ± 1.23	1.31 ± 1.56
WinoGrande	5%	51.38 ± 0.48	50.80 ± 0.12	-0.58 ± 0.46
Five-task mean	5%	40.45 ± 0.27	41.41 ± 0.28	0.96 ± 0.27
PIQA	10%	61.70 ± 1.14	63.60 ± 0.09	1.90 ± 1.05
HellaSwag	10%	29.86 ± 0.05	31.48 ± 0.10	1.62 ± 0.11
ARC-Easy	10%	36.73 ± 0.73	37.53 ± 0.23	0.80 ± 0.96
ARC-Challenge	10%	21.90 ± 0.76	22.64 ± 0.68	0.74 ± 0.96
WinoGrande	10%	50.83 ± 1.19	51.01 ± 0.59	0.18 ± 1.69
Five-task mean	10%	40.20 ± 0.34	41.25 ± 0.13	1.05 ± 0.46
PIQA	15%	61.61 ± 0.72	63.22 ± 0.33	1.61 ± 0.60
HellaSwag	15%	29.42 ± 0.18	31.16 ± 0.26	1.74 ± 0.25
ARC-Easy	15%	36.24 ± 0.56	37.22 ± 0.72	0.98 ± 1.00
ARC-Challenge	15%	22.35 ± 0.48	22.41 ± 0.10	0.06 ± 0.40
WinoGrande	15%	50.62 ± 0.95	50.64 ± 0.66	0.03 ± 0.67
Five-task mean	15%	40.05 ± 0.28	40.93 ± 0.21	0.88 ± 0.08

Table 9: Zero-shot 1B base-checkpoint accuracy across three matched seeds. Values with $\pm$ report mean and standard deviation; Δ is RNN-PPT minus no PPT in percentage points. At 0% noise, entries marked $\approx$ are computed from rounded task means using the PIQA value in Table 7; their across-seed standard deviations are unavailable. ARC-Challenge changes are small relative to seed variation at several noise levels, and WinoGrande is mixed.

C C4 Pre-Pre-Training Across Scales

In the main text we only include the C4-PPT control at 1B scale (Section 4.5). For completeness, Table 10 extends the 160M controlled-noise comparison in Figure 2 to include both Random-PPT and C4-PPT alongside the Baseline, Dyck-PPT, and RNN-PPT methods. All entries are mean $\pm$ standard deviation of final C4 validation loss over three seeds. Lower is better.

Method	0%	10%	30%	50%
Baseline	3.627 ± 0.001	3.659 ± 0.003	3.719 ± 0.003	3.818 ± 0.017
Random PPT	3.621 ± 0.002	3.654 ± 0.004	3.710 ± 0.006	3.792 ± 0.005
Dyck PPT	3.619 ± 0.000	3.645 ± 0.001	3.703 ± 0.003	3.798 ± 0.006
C4 PPT	3.613 ± 0.004	3.635 ± 0.003	3.703 ± 0.005	3.786 ± 0.007
RNN PPT	$\mathbf{3.603 \pm 0.002}$	$\mathbf{3.628 \pm 0.005}$	$\mathbf{3.681 \pm 0.004}$	$\mathbf{3.761 \pm 0.003}$

Table 10: Final C4 validation loss at 160M scale with Random-PPT and C4-PPT included. Values are mean $\pm$ standard deviation across three seeds.

RNN-PPT remains the strongest method at every tested noise level, improving validation loss over C4-PPT by 0.010 at 0% noise and by 0.025 at 50% noise. Random-PPT, by contrast, stays close to the baseline throughout, which is consistent with the main-text interpretation that warm-up alone is not sufficient.

D Effect of Extended PT Budget

We extend the 160M PT budget from 10k to 20k steps at noise rates $\{0\%, 10\%, 30\%, 50\%\}$ and compare RNN-PPT with Baseline models trained fully from scratch. Table 11 reports the final C4 validation loss at 20k PT steps across three seeds. RNN-PPT remains better than the baseline at every tested noise level.

Noise	PT steps	Baseline	RNN-PPT	Δ (Baseline $-$ RNN)
0%	20k	3.506 ± 0.001	3.485 ± 0.001	$+0.021 \pm 0.001$
10%	20k	3.529 ± 0.002	3.508 ± 0.003	$+0.021 \pm 0.004$
30%	20k	3.582 ± 0.002	3.554 ± 0.001	$+0.028 \pm 0.003$
50%	20k	3.656 ± 0.007	3.625 ± 0.002	$+0.031 \pm 0.005$

Table 11: Final C4 validation loss at 160M scale with Baseline and RNN-PPT, reported at PT step 20k as mean $\pm$ standard deviation across three seeds. RNN-PPT remains better than the baseline across all tested noise levels.

E Transfer Across Model Architectures

We evaluate two additional randomly initialized model architectures, Qwen2.5-0.5B [Qwen Team, 2024] and Gated DeltaNet-128M [Yang et al., 2025], on clean and 30%-corrupted C4 PT. Within each architecture, the no-PPT and RNN-PPT runs use matched data, PT budgets, and optimization settings. Table 12 reports the paired reduction in final C4 validation loss, averaged over three seeds. These checks show gains for these two architectures; they do not establish a uniform benefit across model families.

Model architecture	Clean PT	30%-corrupted PT
Qwen2.5-0.5B (Transformer)	$+0.0073 \pm 0.0056$	$+0.0158 \pm 0.0078$
Gated DeltaNet-128M (linear attention)	$+0.0919 \pm 0.0156$	$+0.1244 \pm 0.0204$

Table 12: Paired reduction in final C4 validation loss, $\Delta = L_{\text{no-PPT}} - L_{\text{RNN-PPT}}$, reported as mean $\pm$ standard deviation over three seeds. Positive values favor RNN-PPT.

F Additional PPT Source Controls

We compare synthetic sources using Pythia-160M, 500 PPT steps, 10K C4 PT steps, and three seeds. Unless stated otherwise, each source is compared with matched no-PPT runs under clean and 30%-corrupted PT. We report $\Delta = L_{\text{no-PPT}} - L_{\text{PPT}}$ in final C4 validation loss; positive values favor PPT.

Linear-state and nonlinear RNN controls. For the linear-state controls, we generate tokens from $h_t = W_1 h_{t-1} + \epsilon_t$ and $x_t \sim \text{Categorical}(\text{softmax}(W_2 h_t/\tau))$. The recurrent noisy variant uses $W_1 = \rho Q$ with $\rho = 0.9$ and $\epsilon_t \sim \mathcal{N}(0, (1-\rho^2)I)$, where Q is obtained from a seeded Gaussian matrix by QR decomposition. The non-recurrent variant sets $W_1 = 0$ and samples $h_t = \epsilon_t \sim \mathcal{N}(0, I)$, matching the recurrent noisy variant's stationary state marginal and emission rule. The recurrence-only variant sets $h_t = Qh_{t-1}$ without per-step noise. For the nonlinear RNN controls, we replace the original token-feedback RNN's identity activation with $\tanh$ or ReLU while holding the remaining setup fixed. Table 13 reports both control series. Simple recurrence and nonlinear activation do not reproduce the original RNN's gain under this budget.

PPT source	Clean PT Δ	30%-corrupted PT Δ
Linear-state controls		
Non-recurrent noise ($W_1 = 0$)	$+0.0074 \pm 0.0015$	$+0.0087 \pm 0.0067$
Recurrent + per-step noise	$+0.0049 \pm 0.0032$	$+0.0011 \pm 0.0084$
Recurrent without per-step noise	$+0.0060 \pm 0.0058$	$+0.0013 \pm 0.0013$
Token-feedback RNN variants		
Original linear RNN	$+0.0260 \pm 0.0026$	$+0.0319 \pm 0.0091$
Tanh RNN	$+0.0036 \pm 0.0036$	-0.0004 ± 0.0049
ReLU RNN	$+0.0050 \pm 0.0056$	$+0.0006 \pm 0.0104$

Table 13: Final-loss reduction relative to matched no-PPT runs, reported as mean $\pm$ standard deviation over three seeds. The two blocks are distinct source-control series under the same 160M, 500-step PPT, 10K-step PT protocol.

Random Transformer and Dyck controls. Table 14 compares sequences from randomly initialized Transformers with the RNN and Dyck sources under the same 160M and 500-step PPT setting. Dyck yields smaller positive mean gains, while the random-Transformer source has smaller and more variable effects. Together with the metamer diagnostic in Table 1, these results do not support recurrence or generator complexity alone as sufficient explanations for the RNN-PPT gain.

PPT source	0%	10%	30%	50% PT noise
RNN	$+0.0260 \pm 0.0026$	$+0.0232 \pm 0.0072$	$+0.0319 \pm 0.0091$	$+0.0524 \pm 0.0172$
Dyck	$+0.0074 \pm 0.0011$	$+0.0106 \pm 0.0035$	$+0.0132 \pm 0.0019$	$+0.0201 \pm 0.0183$
Random Transformer	$+0.0067 \pm 0.0024$	$+0.0049 \pm 0.0156$	$+0.0040 \pm 0.0188$	$+0.0152 \pm 0.0275$

Table 14: Paired final-loss reduction relative to no PPT across three seeds (mean $\pm$ standard deviation). Positive values favor the synthetic source.

G Cross-Noise RNN Design Ablations

For readability, Figure 7 in the main text plots the 0% and 30% PT settings. Table 15, Table 16, and Table 17 report the complementary 10% and 50% noise levels for the same three ablation families, so together they cover the full $\{0, 10, 30, 50\}\%$ sweep. Lower is better. The full sweep supports the interpretation in the main text.

H Attention-Probe Protocol and Interpretation

The attention analyses in Section 6.1 use a held-out probe set with independently injected corruption and do not reuse the training batches themselves. This probe uses token-level corruption: within

PT noise	Baseline	RNN-m=1	RNN-m=10	RNN-m=100	RNN-m=500	RNN-m=1000	RNN-m=2000	RNN-m=3000
10%	3.6581	3.6607	3.6449	3.6310	3.6302	**3.6225**	3.6254	3.6370
50%	3.8250	3.8008	3.8143	3.7766	3.7763	**3.7700**	3.7789	3.7917

Table 15: Generator-count ablation at the complementary 10% and 50% PT noise levels. Together with the 0% and 30% curves in Figure 7, these results support the low-bias principle: performance improves once the ensemble reaches roughly 100 generators and is strongest around 1000–3000.

PT noise	Baseline	RNN-h=8	RNN-h=16	RNN-h=32	RNN-h=64	RNN-h=128	RNN-h=256	RNN-h=512	RNN-h=1024
10%	3.6577	3.6290	3.6282	3.6302	**3.6255**	3.6300	3.6301	3.6562	3.6400
50%	3.8367	3.7838	**3.7652**	3.7769	3.7750	3.7753	3.7860	3.8140	3.8143

Table 16: Hidden-size ablation at the complementary 10% and 50% PT noise levels. The full sweep preserves the same broader pattern: generators in a learnable complexity range ($h \in \{16, 32, 64\}$) transfer best, while large generators ($h \in \{512, 1024\}$) lose most of the gain.

each clean C4 validation sequence, we randomly replace the specified fraction of token positions and record those positions as the noisy-token mask. For the head-level heatmap in Figure 10, we apply a single fixed high-noise probe across all compared checkpoints so that the per-head contrasts are measured under the same evaluation condition. This probe corruption rate is 50%.

H.1 Five-Seed Corruption Probes and Attention Interventions

In this comparison, k-Shuffle Dyck is the source denoted by Dyck in the main text: it interleaves k independent balanced bracket sequences and permits crossing dependencies between bracket types. The additional k-Dyck source requires properly nested brackets across all types [Hu et al., 2025]. The tables below use these explicit names to distinguish the two sources.

Checkpoints and validation loss. We use 160M models after 10K PT steps at 30% noise and report means $\pm$ standard deviations over five matched seeds. This follow-up comparison uses five seeds, whereas the main attention figures use three. Table 18 reports the final C4 validation-loss reductions relative to the matched No-PPT models. RNN-PPT has the largest mean reduction among these sources.

Corruption probes. We focus on layers 9–11 because the preceding layerwise analysis places the largest PPT-associated attention differences in the upper layers. At a 50% probe corruption rate, we compare five ways of constructing noise: uniform-vocabulary replacement, replacement using the C4 unigram distribution, replacement with tokens from other C4 sequences, within-sequence permutation, and replacement with contiguous C4 spans. For the first four probes, the clean examples and corruption positions are held fixed while the replacement tokens vary. Table 19 reports the paired change in noise-to-noise attention relative to No-PPT, averaged over layers 9–11. Negative values indicate less attention between corrupted tokens. RNN-PPT shows a negative change in every seed under each of the five probes. The other sources have smaller mean reductions or less consistent signs across probes, indicating that the RNN-PPT pattern extends beyond uniform random replacements.

Intervention protocol. At a fixed checkpoint and input, we multiply attention weights from corrupted query positions to corrupted key positions in layers 9–11 by a factor s and then renormalize each attention row. Thus, $s = 1$ is the unmodified model, $s = 0$ removes this route, $s = 0.5$ weakens it, and $s = 2$ strengthens it before renormalization. We test $0.5\times$ and $2\times$ under uniform-vocabulary and cross-sequence C4 replacement, with an additional $0\times$ ablation for the uniform probe. Table 20 reports changes in clean-token prediction loss relative to the same model without intervention, rather than relative to No-PPT. For all three tested models, every route reduction lowers clean-token loss in all five seeds, and every $2\times$ intervention raises it in all five seeds.

Which predictions change? Table 21 separates clean targets according to whether the immediately preceding query token is corrupted or clean under cross-sequence C4 replacement. The loss changes concentrate on clean targets following corrupted queries, with much smaller changes when

PT noise	Baseline	RNN-V=64	RNN-V=256	RNN-V=1024	RNN-V=4096	RNN-V=16384	RNN-V=50304
10%	3.6577	3.6441	3.6378	3.6364	3.6273	3.6267	**3.6248**
50%	3.8294	3.7998	3.7958	3.7995	3.7801	3.7778	**3.7692**

Table 17: Vocabulary-size ablation at the complementary 10% and 50% PT noise levels. Larger vocabularies are strong overall, consistent with using broad token support to reduce source-specific bias.

PPT source	$L_{\text{No-PPT}} - L_{\text{PPT}}$
RNN	$+0.0278 \pm 0.0113$
k-Dyck	$+0.0108 \pm 0.0075$
k-Shuffle Dyck	$+0.0093 \pm 0.0050$
Random-token	$+0.0087 \pm 0.0058$

Table 18: Final C4 validation-loss reductions in the five-seed mechanism comparison. Positive values favor PPT.

Corruption probe	RNN	k-Shuffle Dyck	k-Dyck	Random-token
Uniform vocabulary	-0.1060 ± 0.0234 (5/5)	-0.0179 ± 0.0220 (4/5)	$+0.0213 \pm 0.0186$ (0/5)	-0.0436 ± 0.0266 (5/5)
C4 unigram	-0.0222 ± 0.0131 (5/5)	-0.0060 ± 0.0099 (4/5)	-0.0078 ± 0.0135 (4/5)	-0.0028 ± 0.0088 (2/5)
Cross-sequence C4	-0.0340 ± 0.0228 (5/5)	-0.0237 ± 0.0070 (5/5)	-0.0081 ± 0.0195 (3/5)	-0.0194 ± 0.0255 (4/5)
Within-sequence	-0.0235 ± 0.0150 (5/5)	-0.0133 ± 0.0043 (5/5)	$+0.0051 \pm 0.0107$ (2/5)	-0.0112 ± 0.0100 (5/5)
Contiguous C4 spans	-0.0999 ± 0.0416 (5/5)	-0.0708 ± 0.0235 (5/5)	$+0.0045 \pm 0.0427$ (2/5)	-0.0169 ± 0.0237 (4/5)

Table 19: Paired changes in noise-to-noise attention relative to No-PPT at 50% probe corruption. Values are mean $\pm$ standard deviation over five matched seeds; parentheses count seeds with a negative change.

Corruption probe	Scale	No-PPT	k-Shuffle Dyck-PPT	RNN-PPT
Uniform vocabulary	$0\times$	-0.1225 ± 0.0439	-0.0387 ± 0.0098	-0.0855 ± 0.0385
	$0.5\times$	-0.0325 ± 0.0124	-0.0093 ± 0.0025	-0.0253 ± 0.0116
	$2\times$	$+0.0327 \pm 0.0126$	$+0.0095 \pm 0.0024$	$+0.0285 \pm 0.0126$
Cross-sequence C4	$0.5\times$	-0.0460 ± 0.0081	-0.0112 ± 0.0028	-0.0299 ± 0.0072
	$2\times$	$+0.0488 \pm 0.0107$	$+0.0121 \pm 0.0029$	$+0.0317 \pm 0.0082$

Table 20: Attention interventions in layers 9–11. Entries are changes in clean-token loss relative to the unmodified checkpoint (mean $\pm$ standard deviation over five seeds); negative values indicate improvement.

the preceding query is clean. This connects the intervention effect to recovering clean text from corrupted context.

Interpretation and scope. Together, the probes and interventions support the noise self-modeling account: RNN-PPT consistently reduces attention between corrupted tokens, and directly weakening this route improves clean-token recovery at fixed checkpoints. These interventions establish that changing this route can affect prediction, but they do not quantify how much of the training-time PPT advantage is mediated by the route. In particular, the Dyck-based sources also improve validation loss despite weaker or less consistent attention changes, suggesting that other source-dependent effects may contribute to PPT transfer.

Clean-token subset	Scale	No-PPT	k-Shuffle Dyck-PPT	RNN-PPT
All clean targets	$0.5\times$	-0.0460 ± 0.0081	-0.0112 ± 0.0028	-0.0299 ± 0.0072
After a corrupted query	$0.5\times$	-0.0905 ± 0.0191	-0.0220 ± 0.0055	-0.0590 ± 0.0148
After a clean query	$0.5\times$	-0.0010 ± 0.0032	-0.0003 ± 0.0003	-0.0006 ± 0.0006
All clean targets	$2\times$	$+0.0488 \pm 0.0107$	$+0.0121 \pm 0.0029$	$+0.0317 \pm 0.0082$
After a corrupted query	$2\times$	$+0.0974 \pm 0.0257$	$+0.0237 \pm 0.0058$	$+0.0627 \pm 0.0169$
After a clean query	$2\times$	-0.0003 ± 0.0045	$+0.0004 \pm 0.0002$	$+0.0004 \pm 0.0008$

Table 21: Clean-token loss changes by target subset under cross-sequence C4 replacement. Each entry is relative to the same unmodified checkpoint (mean $\pm$ standard deviation over five seeds). The query is the token immediately preceding the clean target.

I RNN Generator Architecture

Our RNN-based PPT source is generated by a simple recurrent language model. For vocabulary size V and hidden size H, each generator maintains a hidden state $h_t \in \mathbb{R}^H$ and updates it according to

$$h_t = Ae_{x_{t-1}} + Wh_{t-1} + b, \qquad \ell_t = Ch_t + d, \tag{3}$$

where $e_{x_{t-1}} \in \mathbb{R}^V$ is the one-hot vector of the previous token, $A \in \mathbb{R}^{H\times V}$ is the input projection, $W \in \mathbb{R}^{H\times H}$ is the recurrent matrix, and $C \in \mathbb{R}^{V\times H}$ is the output projection. The next-token distribution is $p_t = \mathrm{softmax}(\ell_t/\tau)$, and the next token is sampled autoregressively as $x_t \sim \mathrm{Categorical}(p_t)$, with temperature $\tau = 1$ in the default setup. The hidden state is initialized to zero, and the first token is sampled uniformly from the vocabulary. We do not sample variable-length sequences: each raw RNN sequence is generated at length 2048 before training-time packing. This choice keeps the synthetic source simple and stable while still producing temporal dependencies through recurrence, autoregressive feedback, and token sampling.

All RNN parameters are sampled once at random and then kept fixed; the generators themselves are never trained. In the implementation, entries of A are sampled i.i.d. from a zero-mean Gaussian with standard deviation $1/\sqrt{V}$, while entries of W and C are sampled i.i.d. with standard deviation $1/\sqrt{H}$. The bias terms b and d are initialized to zero. To keep the dynamics stable while preserving long-range mixing, we rescale W so that its estimated spectral radius, computed by power iteration, is 0.9.

J Compute Budget and Runtime

To contextualize our choice of scale, Table 22 summarizes representative wall-clock runtimes measured on a single NVIDIA H200 GPU. The synthetic PPT stage is lightweight: it typically takes only about 5-10 minutes for 500 steps.

The main PT runs are much more expensive. At 160M scale, a standard 10K-step PT run takes about 1 hour. At 1B scale, the 25K-step PT runs used in Section 4.5 take about 22 hours each. Taking into account the experiments across different random seeds and different noise levels, the aggregate compute budget is substantial.

All synthetic data preparation for Dyck, Random, and RNN PPT sources is performed on CPU and completes within an hour on our 96-core server. These data can be reused across experiments and do not dominate the end-to-end training workflow.

Configuration	Stage	Budget	Approx. wall-clock
160M main setting	synthetic PPT	500 steps	$\sim$5 min
160M main setting	PT	10K steps	$\sim$1 hour
1B scale-up setting	synthetic PPT	1000 steps	$\sim$50 min
1B scale-up setting	PT	25K steps	$\sim$22 hours

Table 22: Representative runtime costs on our hardware.

K FineWeb Quality-Score Partition

For the natural-noise experiment, we begin with a tokenized 3M-document FineWeb subset. We score each document with a frozen external causal language model, `openlm-research/open_llama_3b_v2` [Geng and Liu, 2023], and use the model's mean next-token cross-entropy over the document as a scalar quality score. We choose this reference model for two reasons. First, reference-model-based filtering is a strong baseline for data pruning, and recent evidence shows that even relatively small reference models can identify high-quality subsets that improve downstream pre-training performance [Ankner et al., 2024]. Second, OpenLLaMA-3B-v2 is a different model family from the Pythia architectures used in our main experiments and was trained on a broad public mixture rather than on C4 alone, including Falcon RefinedWeb, StarCoderData, and the Wikipedia/arXiv/books/StackExchange portions of RedPajama, which helps decouple the partition from our downstream training setup.

We sort all documents by this score and retain only the two extremes of the ranking used in the main paper. The high-quality subset is drawn from the bottom third of the score distribution, and the low-quality subset from the top third. The corresponding cutoffs are the 33rd and 67th percentiles of the full distribution. This means the split is fully determined by a fixed external scorer and percentile thresholds, with no manual document selection. To keep the PT budget comparable across subsets, we traverse documents within each retained subset in score order and accumulate them until reaching roughly 660M tokens.

This procedure yields two PT conditions selected by reference loss, with the training recipe otherwise fixed. The following audits assess their differences in natural web noise.

The reference-model scoring step is a one-time preprocessing cost. On a single NVIDIA H200 GPU in bfloat16, scoring 660M tokens with the 3B reference model takes about 4 hours.

Independent audits of the partition. We use two checks to assess whether the higher-reference-loss group contains more natural web noise. First, we pool and shuffle 100 documents from each group, label them with GPT-5.6-Sol using a fixed rubric, and manually verify every label before revealing group membership. Second, we measure rule-based indicators of non-prose structure and malformed text in 50,000 documents per group, adapting indicators from prior web-data audits [Raffel et al., 2020, Penedo et al., 2024, Soldaini et al., 2024]. To control for differences in document length, we match documents with identical token counts, yielding 31,571 pairs. Table 23 reports the results.

Audit indicator	Lower-loss group	Higher-loss group	Difference (pp)
Noisy label	15%	40%	+25
Borderline-or-noisy label	35%	68%	+33
Non-prose structure (length matched)	13.71%	21.29%	+7.58
Malformed text (length matched)	0.31%	0.72%	+0.41

Table 23: Independent audits of the FineWeb partition. Label proportions use 100 documents per group, labeled by an LLM and individually verified by a human before unblinding. Rule-based indicators use 31,571 pairs matched by token count. Differences are higher-loss minus lower-loss proportions in percentage points.

The higher-loss group contains more documents labeled as noisy (40% versus 15%; Fisher's exact test, $p = 1.20 \times 10^{-4}$). Verified examples include navigation, boilerplate, fragments, advertising/SEO content, and incoherent grammar. The higher rates of non-prose structure and malformed text after length matching support the same conclusion: the higher-loss group contains more natural web noise.

L Representative Cases of Naturally Clean and Noisy Data

Table 24 shows representative cases from the actual FineWeb subsets used in Section 4.4. We include two examples from the retained high-quality subset and two from the retained low-quality subset. The cleaner subset more often contains coherent article-style prose, while the noisier subset more often contains templated listings, repetitive boilerplate, and ad-like fragments.

In pre-training pipelines, these templated listings, directory dumps, SEO boilerplate, and ad-like fragments are precisely the kinds of content that are regarded as noise and routinely filtered out in practice [Penedo et al., 2024]. Prior empirical studies likewise show that noisy or low-quality data can weaken pre-training outcomes, and that filtering decisions involve tradeoffs between quality and coverage [Zhang et al., 2026, Longpre et al., 2023]. Their defining limitation is therefore not the absence of surface structure, but the scarcity of transferable linguistic signal: although they can contain repeated templates and other local surface regularities, they contribute little generalizable information. From this perspective, this partition captures a realistic and practically relevant form of noise that models encounter at web scale.

Subset	Example type	Case
Cleaner	Entertainment article	If you haven't seen the new trailer for The Lego Batman Movie, then you are missing out on one of the best laughs you will have all day. Not only is Lego Batman too good to be a side character in The Lego Movie, but Lego Batman rightfully deserves his own film. ...
Cleaner	News report	CONWAY, Ark. (AP) – LaQuentin Miles scored 20 points, including 10 of Central Arkansas' final 11 points, and the Bears held on to beat Southern Illinois-Edwardsville 80-78 on Saturday. Jarvis Garner added 19 points and Robert Crawford scored 18 for Central Arkansas. ...
Noisier	Directory / ad mix	IFSC code: ICIC0002980 and MICR code: 342229203; ICICI BANK DANWARA address: Icici Bank Ltd., Village Danwara, Pin - 342037, Tehsil Baori, Dist. Jodhpur, Rajasthan; Branch code is 002980. Credit Score of 750 = Easy approval. ...
Noisier	Listing / boilerplate	Use Cvent to book the G Casino Sheffield in Sheffield, England for your event and get a great rate. Find upcoming events at Grosvenor G Casino Sheffield in Sheffield. Full event details plus travel info, opening times + venue info. This table is no longer available, followed by unrelated multilingual boilerplate. ...

Table 24: Representative cases from the FineWeb subsets used in the natural-noise experiment. Snippets are abbreviated for readability.

M Unigram, Bigram, and Trigram Metamer Construction Details

This appendix provides the construction details for the unigram, bigram, and trigram metamer diagnostic reported in Section 6.2 and Table 1.

RNN-subset. The main-paper RNN-PPT source is generated by an ensemble of 1000 randomly initialized RNNs. Fitting a full-vocabulary ($V = 50{,}304$) trigram count table over the entire source is storage-prohibitive, so we first construct an "RNN-subset" by uniformly sampling sequences from the full 1000-generator source. The subset preserves the original generator-level diversity. Its size is set so that a 500-step PPT run at the main-paper batch size and sequence length (effective batch 32, sequence length 2048) consumes strictly less than one epoch of the subset; consequently, training never revisits a token.

Metamer fitting and sampling. For each order $n \in \{1, 2, 3\}$, we fit the corresponding model on exactly the same RNN-subset, so that the metamer statistics come from the same token distribution as the reference condition. We then sample new token sequences autoregressively from the fitted model, drawing a fresh stream whose total token count matches the PPT token budget used by every other PPT condition. Unigram, bigram, trigram, and RNN-subset conditions therefore all see the same number of PPT tokens with no token repetition during training. For trigram sampling, each sequence is initialized from an observed start bigram. At generation time, if the current bigram context has no observed trigram continuation, we back off to the empirical bigram model conditioned on the last token; if that context also has no observed continuation, we sample from the empirical unigram distribution. Bigram sampling uses the analogous unigram back-off.

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 claims are stated in the abstract and Section 1, and are supported by the experiments in Sections 4.2–4.5.

   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: Section 7 discusses limitations such as model scale, PT budget, source selection cost, and the limited range of tested generators and unresolved sequence statistics responsible for transfer.

   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: [N/A]

Justification: The paper does not present theoretical results requiring formal proofs.

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 4.1 and Appendix A provide the main datasets, model settings, hyperparameters, and evaluation details.

   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: [Yes]

Justification: The datasets and models used are public. The code repository, with instructions for data preparation and reproducing the main C4 experiments, is linked at the end of the abstract.

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 4.1 and Appendix A give the dataset, model, optimizer, hyperparameter, and evaluation details.

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: Main experimental results are averaged over three seeds; the additional mechanistic probes and attention interventions use five matched seeds. Standard deviations or error bars are reported in the corresponding figures and tables.

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: Appendix J reports representative runtimes on a single NVIDIA H200 GPU, and Appendix A gives the training budgets.

   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 work uses public datasets/models and synthetic generators, and does not involve human subjects.

   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: [No]

    Justification: The paper is mainly foundational and does not include a dedicated discussion of both positive and negative societal impacts.

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 submission does not release a pretrained model, image generator, scraped dataset, or other high-risk artifact.

    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: All the assets in this work, either are original or publicly available (e.g., data, models).

    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: [Yes]

    Justification: The released code and its documentation are available in the repository linked at the end of the abstract, including installation, data preparation, and experiment commands.

    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 research does not involve crowdsourcing experiments or 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 research does not involve crowdsourcing or 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: [Yes]

    Justification: We pre-train Pythia-architecture LLMs (160M and 1B parameters) in the main experiments (Section 4) and evaluate two additional randomly initialized architectures in Appendix E. We also use a frozen external LLM, `openlm-research/open_llama_3b_v2`, as a reference model for the FineWeb quality-score partition (Appendix K).

    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.