Training Language Models via Neural Cellular Automata

Dan Lee[*]
UIUC
dhl9@illinois.edu

Seungwook Han[*]
MIT, Improbable AI Lab
swhan@mit.edu

Akarsh Kumar
MIT

Pulkit Agrawal
MIT, Improbable AI Lab

Abstract

Pre-training is crucial for large language models (LLMs), as it is when most representations and capabilities are acquired. However, natural language pre-training has problems: high-quality text is finite, contains human biases, and entangles knowledge with reasoning. This raises a fundamental question: is natural language the only path to intelligence? We propose using neural cellular automata (NCA) to generate synthetic, non-linguistic data for *pre-pre-training* LLMs–training on synthetic-then-natural language. NCA data exhibits rich spatiotemporal structure and statistics resembling natural language while being controllable and cheap to generate at scale. We find that pre-pre-training on only 164M NCA tokens improves downstream language modeling by up to 6% and accelerates convergence by up to 1.6× in 400M-3B models trained with Chinchilla optimal data budgets. *Surprisingly, this even outperforms pre-pre-training on 1.6B tokens of high-quality natural language data with more compute.* Investigating what drives transfer, we find that attention layers are the most transferable, and that optimal NCA complexity varies by domain: code benefits from simpler dynamics, while math and web text favor more complex ones. These results enable systematic tuning of the synthetic distribution to target domains. More broadly, our work opens a path toward more efficient models with fully synthetic pre-training.

1 Introduction

Scale has transformed neural networks, enabling emergent abilities like reasoning [39, 41, 3] and in-context learning [11, 79, 86] in large language models (LLMs). However, neural scaling laws predict that continued improvements require exponentially more data [42], which is nearing exhaustion by 2028 [76]. Also, natural language inherits undesirable human biases and needs meticulous data curation before it is used for training [33, 2]. This raises a fundamental question: **Is natural language the only path to learning useful representations?** In this paper, we explore an alternative path using synthetic data from cellular automata.

Our core hypothesis is that the efficacy of natural language for pre-training stems from its underlying structure, rather than its semantics. Text is a lossy record of human cognition and the world it describes, containing diverse kinds of structure, from reasoning traces to procedural instructions [66, 67, 16, 22]. Next-token prediction on such data pressures models to internalize the latent computational processes for coherent continuations, fostering key capabilities of intelligence [21, 41].

[*]Equal contribution.

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

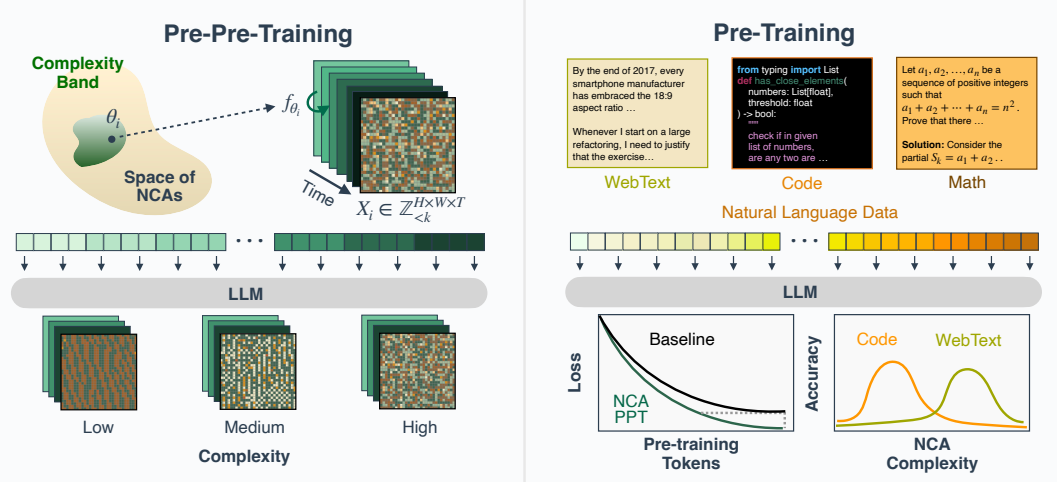


Figure 1: **Overview of NCA Pre-pre-training to Language Pre-training.** We pre-pre-train a transformer with next-token prediction on the dynamics of neural cellular automata (NCA) sampled from selected complexity regions. We then conduct standard pre-training on natural language corpora. NCA pre-pre-training improves both validation perplexity and convergence speed on language pre-training. Interestingly, the optimal NCA distribution varies by downstream domain.

If the key ingredient is exposure to various structures rather than language semantics, then richly structured non-linguistic data could also be effective for language pre-training. To investigate this hypothesis, we employ algorithmically generated data from neural cellular automata (NCA) [56] as a synthetic training substrate. NCA generalize systems like Conway's Game of Life [29] by replacing fixed dynamics rules with neural networks and can generate diverse data distributions with simple local rules. This produces long-range spatio-temporal patterns (see Figure 1) of arbitrary sizes that exhibit heavy-tailed, Zipfian token distributions (see Figure 7 in Appendix B) reminiscent of natural data. To systematically control this distribution, we propose a simple method to compute and sample the desired complexity of NCA tuned for the target domain.

We adopt a pre-pre-training framework[37, 41, 10]: an initial phase of training on NCA dynamics that precedes standard pre-training on natural language. Our ultimate vision is to pre-train entirely on clean synthetic data, followed by fine-tuning on a limited and curated corpora of natural language to acquire semantics [33]. The pre-pre-training framework serves as an early prototype of this paradigm, allowing us to measure how computational primitives learned from synthetic NCA transfer to language tasks. Our contributions are as follows:

1. **A fully synthetic pre-pre-training substrate that transfers to language modeling.** We propose neural cellular automata (NCA) as a fully algorithmic, non-linguistic data source for pre-pre-training. NCA pre-pre-training improves language modeling perplexity by up to 6% and convergence speed by up to $1.6\times$ across web text, math, and code with 400M-3B models trained up to Chinchilla optimal token budgets. These performance gains translate to modest but statistically significant improvements on reasoning benchmarks, such as GSM8K and BigBench-Lite. Most surprisingly, NCA outperforms both web text (C4) and curated natural language data (FineWeb-Edu) as a pre-pre-training source.
2. **Synthetic pre-training enables domain-targeted data design.** We find that the optimal NCA complexity regime varies by downstream task: code benefits from lower-complexity rules while math and web text benefit from higher-complexity ones. NCAs' parametric structure offers a new lever to tune the complexity of training distributions to match the computational character of target domains.
3. **Attention captures the most transferable priors.** The attention layers capture the most useful computational primitives, accounting for the majority of the transfer gains. Attention appears to be the carrier of transferable capabilities such as long-range dependency tracking and in-context learning, whereas MLPs may encode more domain-specific knowledge–making MLP transfer conditional on alignment between the synthetic and target domains.

We further illustrate that models evidently learn different attention behaviors with and without NCA pre-pre-training.

2 Related Works

Synthetic data include outputs generated by LLMs [59, 77, 84, 58, 49, 53, 78] and by simple algorithms. In this work, we pursue the latter, the non-linguistic approach.

Algorithm-Based Synthetic Data In vision, prior work used simple algorithmic procedures (e.g., OpenGL shader images, wavelet models, fractals) to generate synthetic training data [6, 5, 43]. Despite lacking semantic content, these models learn representations that transfer well to real images. Baradad Jurjo et al. [6] argue that what matters is reproducing the statistical structure of the natural world, such as the approximate scale-invariance [27] or the Zipfian distribution [87, 12]. In the language domain, using algorithmically generated data is less common, but have been explored [68, 23, 61]. Chiang & Lee [17], Hu et al. [36] showed that pre-training LLMs on synthetic data generated by context-free grammars can be useful for language modeling. Bloem [10] shows empirically that pre-training models on data generated by randomly initialized LSTMs can improve convergence speed for fine-tuning. Theoretically, they frame pre-training as an approximation of Solomonoff induction and demonstrate that data passed through computational processes can create depth that primes models for "universal" pattern recognition. Similarly, Shinnick et al. [71] showed that warming up language models by training on abstract algorithmic sequences and formal languages like Dyck languages instill non-trivial structure in MLP and attention layers, accelerating the acquisition of skills like long-context recall. Berkovich & Buehler [8] and Berkovich et al. [9] trained LLMs on 1D-elementary cellular automata, but did not study the usefulness of the learned representations for language. The key shortcoming of prior approaches with formal languages and random strings is that they are neither diverse nor controllable, offering no principled way to expand the space of tasks or to dial complexity up and down. NCAs address this gap. The parametric structure of NCA yields diverse dynamics and allows systematic control over complexity. This enables us to ask not only whether synthetic data can transfer, but what structural properties make it effective.

Shared Underlying Computation A growing body of work suggests that neural networks learn general computations that transfer across domains, raising the possibility that synthetic algorithmic data could instill such primitives directly. Lu et al. [52] show that LLMs trained on natural language can transfer to seemingly unrelated domains like vision and protein folding. Huh et al. [38] illustrate that foundation models across different modalities are converging in representation, hinting that they learn a common structure. Going further, Mirchandani et al. [54] show that LLMs already have in-context learning capabilities for symbolic reasoning, numeric pattern continuation, and robotic control. These works cast LLMs as universal computation engines [52, 64, 85].

Emergent Complexity A central puzzle for algorithmic synthetic data is how simple procedures can give rise to data with rich structure, resembling the complexity of the real world. Researchers have developed various measures to quantify such complexity [50, 55], including Kolmogorov complexity [45], sophistication [57], and assembly index [69]. More recently, epiplexity was introduced as a complexity metric for computationally bounded observers [28], demonstrating that simple deterministic rules can produce data useful for learning good representations. These works suggest that NCAs, despite having simple local rules, can generate arbitrarily complex structures when rolled out over long time horizons, making them a promising source of synthetic training data.

3 Method

We study whether neural cellular automata (NCA) can create training data that teaches transferable computational priors useful for language modeling.

3.1 Neural Cellular Automata (NCA)

NCAs generalize classical cellular automata such as Conway's Game of Life [80]. In the discrete setting used here, a grid evolves by synchronously updating each cell from its local neighborhood at

the previous timestep. A small neural network parameterizes the shared update rule, allowing diverse rules to be sampled rather than hand-designed. Iterating a rule can produce structured spatiotemporal patterns (Figure 1 bottom left).

Random Discrete NCA. We use 2D discrete NCA on a 12×12 grid g with periodic boundaries and state alphabet $n = 10$, where each cell is represented as a 10-dimensional one-hot vector. The transition dynamics are governed by a neural network f_θ that maps each cell's 3×3 neighborhood to logits over next-cell states: $c_i^{(t+1)} \sim \mathrm{softmax}\Big(f_\theta\Big(N(c_i^{(t)})\Big)/\tau\Big)$, where $c_i^{(t)}$ is the state of cell i at time t, $N(c)$ denotes its neighborhood of the given cell, and $\tau = 1 \times 10^{-3}$ is the sampling temperature to introduce small stochasticity. We parameterize f_θ as a 3×3 convolution (4 channels) followed by a cell-wise MLP with hidden size 16 and ReLU activation.

Complexity-based sampling. We outline our algorithm to generate diverse training data in Algorithm 1 of Appendix A. For each sequence, we randomly initialize the parameters θ of the transition network and sample the initial grid $g^{(0)} \overset{\text{iid}}{\sim} \mathrm{Unif}\{0,\dots,9\}^{12\times12}$. These transition networks f_θ are not learned; they are frozen and unique dynamics functions for each rollout. This procedure yields a distribution over NCA dynamics ranging from trivially predictable states to highly chaotic ones.

To filter trajectories by structural complexity, we serialize each sequence as bytes and compute its gzip compressed-size ratio, $r = \texttt{compressed bytes}/\texttt{raw bytes} * 100$. We sample NCAs with trajectories of $r > 50\%$. gzip under the hood combines Lempel-Ziv compression [88] with Huffman coding. Since Lempel-Ziv compression provides a computable upper bound on Kolmogorov complexity [48], gzip serves as a practical measure of sequence complexity: lower values indicate simpler, more predictable structure, whereas higher values indicate less compressible, more chaotic structure (see Figure 1).

Tokenization. Given an NCA dynamics function f_θ and initial grid $g^{(0)}$, we generate a trajectory $(g^{(0)}, g^{(1)}, \dots, g^{(T)})$, then tokenize it patch-wise as in vision transformers [25]. We partition each grid $g^{(t)}$ into non-overlapping 2×2 patches: $x_0^{(t)}, \dots, x_{P-1}^{(t)}$. With 10 possible states per cell and 4 cells per patch, there are 10^4 distinct patches, which we map bijectively to a vocabulary of 10,000 tokens (excluding special tokens). We serialize the $P = 36$ patches of each grid in row-major order, wrap them with `<grid>` and `</grid>` delimiters, and concatenate across timesteps to form the final sequence $\mathbf{x} = (\texttt{<grid>}, x_0^{(0)}, \dots, x_{P-1}^{(0)}, \texttt{</grid>}, \texttt{<grid>}, x_0^{(1)}, \dots)$ capped at 2048 tokens.

3.2 Training Objective and Interpretation

We train a transformer autoregressively on the tokenized trajectory $\mathbf{x} = (x_1, \dots, x_N)$ using cross-entropy loss: $\mathcal{L} = -\sum_{i=1}^{N} \log p_\phi(x_i \mid x_{<i})$. Because each sequence is generated by a separately sampled θ, next-token prediction encourages the model to infer and apply the trajectory's latent transition rule from context. This setup aligns with Bayesian accounts of in-context learning [34, 83].

4 Experimental Setup

4.1 Training Paradigm

We adopt a three-stage training paradigm [70, 37, 10]:

1. **Pre-pre-training:** An initial training phase designed to instill transferable computational priors before the main pre-training stage. In this work, we propose using synthetic, non-linguistic data (NCA trajectories) for pre-pre-training.
2. **Pre-training:** Standard large-scale training on natural language corpora (web text, code, or math) to acquire linguistic knowledge.
3. **Fine-tuning:** Task-specific adaptation (e.g., instruction tuning).

Our main focus is the transfer from stage 1 to stage 2—whether computational structure learned from synthetic data can accelerate and improve language model pre-training. As a secondary study, we examine stage 3, assessing how these gains carry over to downstream reasoning benchmarks.

4.2 Setup

We generate NCA data by randomly sampling neural network weights that define the transition rule. Each trajectory is thus produced by a unique rule, ensuring diversity across the training distribution. We conduct most of the main experiments with a 1.6B Llama-based transformer [73] and scale up to 3B parameters.

We train on 164M NCA tokens sampled at the $50\%+$ gzip compressibility band, unless otherwise noted. We measure transfer by pre-training on four downstream corpora: OpenWebText [32], OpenWebMath [62], CodeParrot [75], and FineWeb-Edu [51], scaling up to the Chinchilla-optimal regime ($20\times$ the number of params) [35]. We transfer all model weights except the embedding and output layers. All parameters are updated during pre-training. Additional details are in Appendix C.

For downstream reasoning benchmarks, we evaluate on GSM8K [18], HumanEval [14], and BigBench-Lite [72]. We fine-tune the models on the train set for instruction following for GSM8K and BigBench-Lite. We provide the fine-tuning details in Appendix D.

4.3 Baselines

We compare four baseline conditions: (i) **scratch**, in which the model is randomly initialized and trained directly on the language corpus; (ii) **Dyck pre-pre-training**, using the k-Dyck formal language studied by Hu et al. [37]; (iii) **C4 pre-pre-training**; and (iv) **FineWeb-Edu pre-pre-training**. For Dyck, we use $k = 128$ and unbounded potential depth. Within each training stage, we match token counts, optimizer steps, and total FLOPs across conditions; only the pre-pre-training data source varies.

4.4 Hyperparameters

For both our pre-pre-training and baseline runs, we perform a grid search over learning rate and weight decay and use the best hyperparameters for each method. For pre-training, we train four separate models for a single epoch on each dataset: Fineweb-Edu (32B and 98B tokens), OpenWebText (9B tokens), OpenWebMath (4B tokens), and CodeParrot (13B tokens). We report the detailed hyperparameters in Table 1 in Appendix C.

For downstream reasoning benchmarks, we use models pre-trained on OpenWebText, OpenWebMath, and CodeParrot for Big-Bench-Lite, GSM8K, and HumanEval, respectively: matching the benchmark domain approximately to the pre-training domain. We report details on fine-tuning in Appendix D.

4.5 Evaluation Metrics for Transfer

We evaluate transfer from pre-pre-training to pre-training using validation perplexity on a held-out set and convergence speed [10, 37, 42]. Convergence speed is measured as the ratio of tokens to reach a base model's final perplexity relative to the tokens required by the NCA pre-pre-trained model, including any pre-pre-training tokens. Adapting Hu et al. [37], we compute this as, $S^m_{\text{NCA}} = \frac{T^m_{\text{PPT}}+T^m_{\text{PT}}}{T^{\text{NCA}}_{\text{PPT}}+T^{\text{NCA}}_{\text{PT}}}$, where T^m_{PPT} and T^m_{PT} are the pre-pre-training and pre-training tokens that model m needs to reach the final loss. The baseline m is typically the scratch model, for which $T^{\text{scratch}}_{\text{PPT}} = 0$.

On downstream reasoning tasks, our primary evaluation metric is pass accuracy with multiple decodings or pass@k [15]. For BigBench tasks, we compute accuracy on a normalized basis to adjust for random guessing associated with multiple-choice questions [72].

5 Results

We present results on the impact of NCA pre-pre-training on downstream language modeling, and analyze how transfer varies across scale and data complexity.

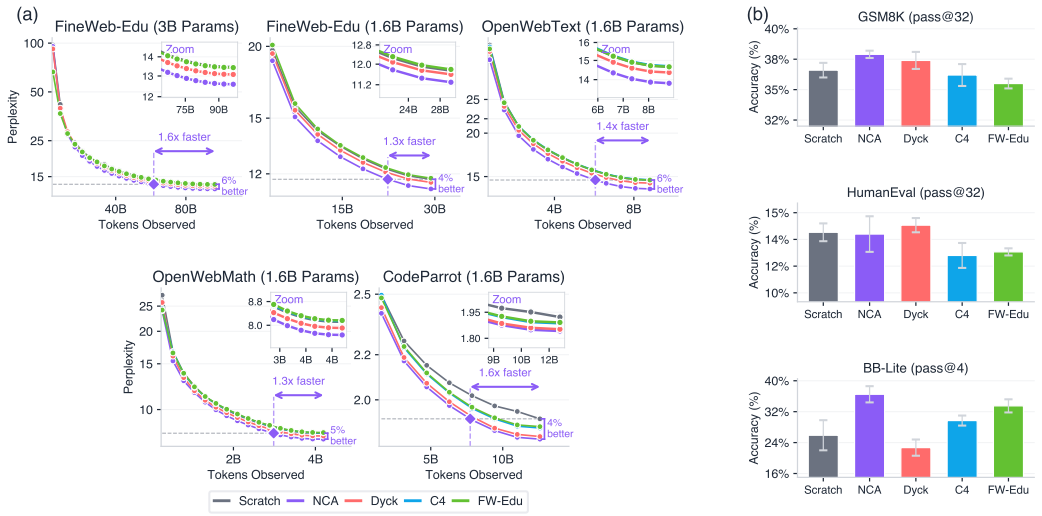


Figure 2: **(a) NCA pre-pre-training improves and accelerates LM pre-training across domains.** Validation perplexity for 3B models on FineWeb-Edu and 1.6B models on FinewebEdu, OpenWebText, OpenWebMath, and CodeParrot. NCA outperforms scratch, Dyck, and even C4 and FineWeb-Edu baselines, yielding 1.3–1.6x faster convergence and 4–6% lower final perplexity. **(b) NCA pre-pre-training provides modest but statistically significant improvement on GSM8k and Bigbench.** Pass@k accuracy and standard deviation error bars for 1.6B language models on reasoning benchmarks.

5.1 NCA Pre-pre-training Improves Language Modeling

We compare the language modeling performance against three baselines: (i) no pre-pre-training ("scratch"), (ii) pre-pre-training on another synthetic language Dyck [37], and (iii) pre-pre-training on natural language datasets C4 and FineWeb-Edu.

As seen in Figure 3b, across model scales (400M, 600M, 1.6B, 3B) and multiple random seeds, NCA pre-pre-trained models consistently outperform all four baselines (Scratch, Dyck, C4 and FineWeb-Edu). On OpenWebText, the best-performing NCA pre-pre-trained 400M model improves downstream perplexity upon the scratch baseline by **8.6%**, and the 3B model improves by **5.8%**. On FineWeb-Edu, the gains grow with scale: moving from 1.6B to 3B (with data budget scaled accordingly), perplexity improvement increases from 4% to 6% and convergence speedup from 1.3x to 1.6x. Figure 3c shows that for the 3B model trained on 98B tokens, the NCA model's convergence advantage over the scratch baseline grows with the token budget, and the perplexity improvement is maintained throughout training. This suggests that NCA's benefits are not fragile artifacts of initialization but persist across model and data scale.

The most surprising observation is that ***pre-pre-training on NCA outperforms pre-pre-training on natural language (C4 and FineWeb-Edu) under matched token and compute budgets*** (Figure 2). We strengthen this comparison by giving the FineWeb-Edu baseline substantially more data—1.6B versus 164M tokens—and retaining its trained embeddings during transfer (Figure 3a). Even with these advantages, NCA models improve upon the FineWeb-Edu baseline by 5% on perplexity and $1.4\times$ on training convergence.

Transfer also extends to math and code (Figure 2a): NCA pre-pre-training reduces validation perplexity on OpenWebMath and CodeParrot by 4–5% and yields up to a $1.6\times$ convergence speedup. The advantage persists and grows during training, indicating greater token efficiency rather than a transient initialization effect.

5.2 Impact of NCA Pre-Pre-Training on Downstream Language and Reasoning Tasks

While perplexity and convergence speed are robust metrics for language modeling, we further evaluate whether the improvements from the NCA pre-pre-training manifest in downstream reasoning. We present the results on GSM8K, HumanEval, and BigBench-Lite in Figure 2b. Given the model scale (1.6B) and limited downstream data budget, absolute performance remains limited; however, we

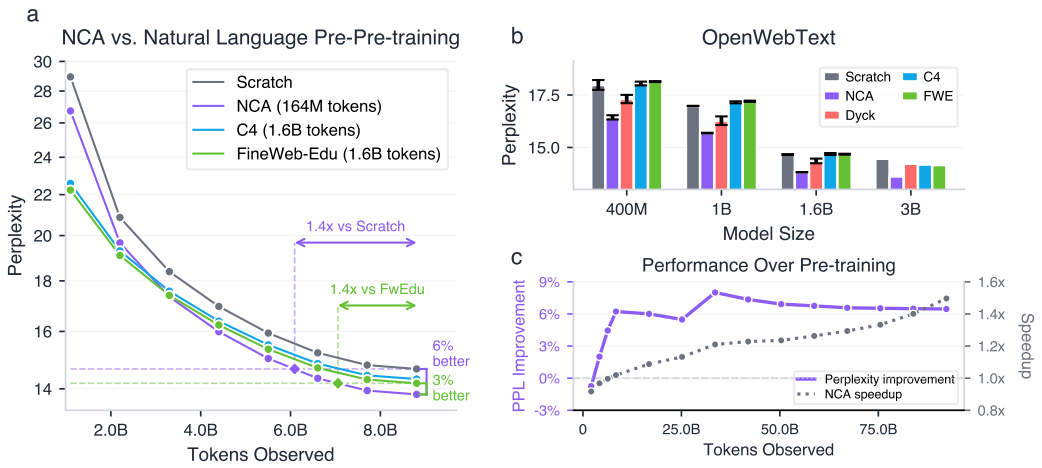


Figure 3: **(a) 164M tokens of NCA pre-pre-training improves convergence speed and perplexity versus 1.6B of natural language pre-pre-training.** Comparison of convergence and final validation perplexity on OpenWebText of models pre-pretrained on 164M tokens of NCA with reinitialized embeddings versus models pre-pretrained on 1.6B tokens on C4 and FineWeb-Edu with embeddings transferred. **(b) NCA pre-pre-training improvements persist across model scale.** Final validation perplexity and standard deviations (seeds) across all baselines and model scales from 400M to 3B. **(c) Perplexity gains remain stable and convergence speedup grows throughout pre-training.** We compute the perplexity improvement and convergence speedup for NCA pre-pretrained models versus scratch models (3B) during 98B tokens of Fineweb-Edu pre-training.

observe **statistically significant improvements** ($p < 0.05$) on GSM8K and Bigbench-Lite compared to the scratch and natural language baselines.

On GSM8K, NCA pre-pre-training improves accuracy from 36.6% to 37.9% at pass@32. On HumanEval, NCA pre-pre-training improves pass@1 but the advantage is not statistically significant. Interestingly, Dyck pre-pre-training is competitive with NCA on HumanEval at higher pass@k. This likely reflects the structural overlap between Dyck languages and code, both of which require tracking nesting logic and delimiter patterns. On BigBench-Lite, NCA pre-pre-training outperforms markedly at higher k, reaching 36.5% at pass@4 compared to 29.7% for C4 and 25.9% for the scratch baseline. In Appendix G, we provide additional experiments for evaluations on the Mostly Basic Python Programs (MBPP) programming benchmark [4], where we observe that NCA models improve upon the scratch baseline by 4.55% at *pass@32*. Together, these results suggest that NCA pre-pre-training can offer modest gains in downstream reasoning, but we leave a more thorough characterization of when and why these gains emerge to future work. A detailed comparison over different pass@k (Table 2) and full t-tests and p-values (Table 3) are reported in Appendix E.

5.3 What Drives Transfer?

The preceding results establish that NCA pre-pre-training improves both language modeling and reasoning. We now investigate the mechanism: which model components capture the transferable structure, and what properties of the synthetic data control transfer effectiveness?

5.3.1 Attention captures the most transferable primitives.

To isolate which model components carry the transfer signal, we selectively re-initialize subsets of weights after NCA pre-pre-training and measure the impact on language modeling. As shown in Figure 4a, re-initializing attention weights causes the largest degradation in transfer across all configurations, indicating that attention captures the most transferable computational primitives.

Other components are more domain-dependent. On OpenWebText, retaining MLP and LayerNorm weights hurts transfer, suggesting that they encode NCA-specific statistics that interfere with language

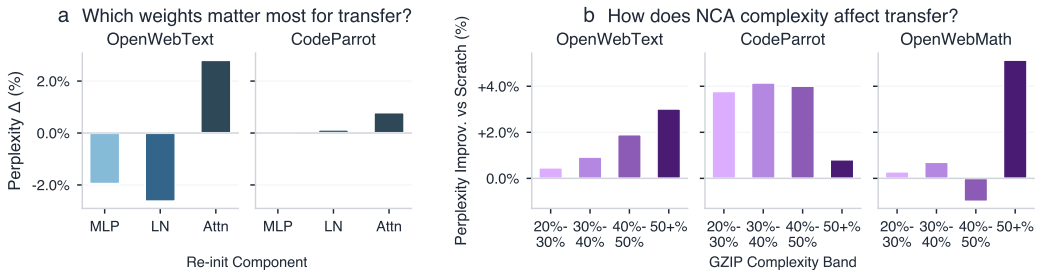


Figure 4: Across ablations, we report the validation perplexity change (more negative is better) on Openwebtext relative to (a) NCA pre-pretrained models with no reinitializations and (b) scratch model. **(a) Attention weights are most crucial for positive transfer.** Re-initializing attention causes the largest degradation across both OpenWebText and CodeParrot, while MLP and LayerNorm effects are domain-dependent. **(b) Optimal NCA complexity is domain-dependent.** Text benefits from higher-complexity data (50%+), while Code peaks at intermediate complexity (30–40%). This suggests that matching synthetic data complexity to the target domain is needed to maximize transfer.

learning. On CodeParrot, retaining these weights has negligible effect. Thus, attention transfers across the tested domains, whereas MLP and LayerNorm transfer depends on source-to-target alignment.

This interpretation agrees with concurrent work locating transferable structure primarily in attention during synthetic pre-training [70, 71]. It is also consistent with analyses of Mixture-of-Experts architectures suggesting that MLP reinforces knowledge than reasoning [40]. Together, these findings suggest a functional division: attention learns general mechanisms for tracking dependencies and inferring latent rules, whereas MLPs store more domain-specific patterns and statistics.



Figure 5: Across repeat lengths, **NCA pre-pre-trained models contain substantially more induction-like heads** than randomly initialized models. We plot the means and ± 1 standard deviation across four model seeds.

To test whether NCA pre-pre-training strengthens a specific in-context mechanism, we probe the prefix-matching component of induction heads [60]. NCA-pre-trained models contain $22\times$ as many above-null prefix-matching heads as randomly initialized models (Figure 5). This provides mechanistic evidence that NCA pre-pre-training places a canonical in-context computation primitive in the weights rather than merely providing a favorable optimization warmup. Appendix H details the probe. To test whether the broader transfer gains can be explained solely by improved initialization statistics, we also compare against first-order-statistics-based mimetic initialization. NCA pre-pre-training outperforms this control (Appendix J), indicating that its benefits extend beyond initialization alone.

5.3.2 Data complexity modulates transfer and the optimum is domain-dependent.

Having established that attention carries the most transferable signal, we next ask: what properties of NCA data affect the transfer? We analyze complexity along two complementary axes: gzip compressibility (as an upper bound to Kolmogorov complexity) and alphabet size n (size of the state space).

Complexity via gzip. We generate NCA trajectories of varying gzip compressibility bands (20–30%, 30–40%, 40–50%, 50%+). Smaller compression ratios imply regular, low-entropy dynamics that are more predictable, whereas larger compression ratios generate more diverse and unpredictable trajectories with richer spatiotemporal structure, as illustrated in Figure 1.

As shown in Figure 4b, the optimal complexity band varies by downstream domain. OpenWebText benefits from less compressible (more complex) NCA rules in the 50%+ band, while CodeParrot shows a sweet spot at intermediate complexity (30–40% gzip). Strikingly, this aligns with the intrinsic complexity of the target corpora themselves in Figure 7 of Appendix B: OpenWebText and OpenWebMath have gzip ratios of 60–70%, whereas CodeParrot is substantially more compressible at 32%. The correlation is somewhat direct. Domains with higher intrinsic complexity benefit from higher-complexity synthetic data, and vice versa. This suggests a plausible and practical principle: **matching the complexity of synthetic pre-training data to the target domain maximizes transfer**.

Rule space expressiveness via alphabet size. We vary the NCA state alphabet $n \in \{2, 10, 15\}$, which controls the diversity of possible local interactions. As shown in Figure 6, larger alphabets ($n = 10, 15$) exhibit diminishing returns: performance improves the most at an intermediate NCA token budget, then the improvement gap narrows. Surprisingly, the smallest alphabet ($n = 2$) scales most favorably, continuing to improve where larger alphabets plateau.

As seen in Figure 8 in Appendix B, when increasing n, the resulting NCA data naturally becomes more complex. This result suggests that although larger rule spaces can express more complex dynamics, better guidance is necessary to sample a diverse set of NCAs that optimally transfer to language. Thus, constraining the space to n=2 may paradoxically help by concentrating samples on dynamics with more consistent and transferable structure.

These results reveal that it is not simply "more NCA data is better." The complexity of synthetic data, both gzip and alphabet size, must be calibrated. This offers a lever unavailable in natural language pre-training: the ability to tune the training distribution to match the computational character of target domains.

We further discuss the implications for domain-targeted pre-training in Section 6.

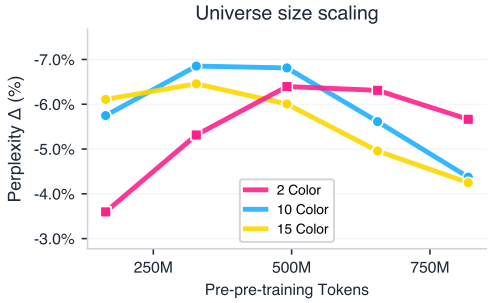


Figure 6: **NCA alphabet size changes scaling behavior.** Smaller alphabets scale better.

6 Discussion

Why should we expect transfer? NCA data are substantially different from natural language and are generated by deterministic processes, prompting the question of why one should expect transfer at all? Yet our results from Section 5 and Figure 11 in Appendix I shows transfer occurs in both directions during training—NCA training improves language loss and vice versa, suggesting a shared underlying structure.

NCAs can provide a direct signal for in-context rule inference.. In natural language, models can exploit semantic shortcuts or co-occurrence priors [1, 31]. In contrast, each NCA sequence instead follows a hidden transition rule that must be inferred from context [44], without any semantic knowledge to fall back on. Such rule inference is also central to modeling language [11, 79, 24], math, and code [83, 30, 19]. Because NCAs encompass a broad, universal class of computable functions [20], some of which realize Turing-complete systems [65, 81], their diversity makes memorization impractical and favors general rule-inference mechanisms [46]. This framing is supported by our findings in Section 5.3.1 that attention carries the most transferable structure. Training on NCA could help with the formation of induction heads (copy-and-apply behavior), which Olsson et al. [60] identify as the important substrate for in-context learning. A complementary motivation comes from *epiplexity* [28]: although classical information theory suggests deterministic transformations cannot increase information content [63], this assumes a computationally unbounded observer. For computationally bounded observers, Finzi et al. [28] show that simple local rules give rise to emergent structures (e.g., gliders, collisions) that cannot be brute-force simulated within a finite model's capacity. This forces the model to learn coarser-grained abstractions, mirroring the representational pressure that drives language modeling.

Why are 164M tokens of automata better than 1.6B tokens of text? We propose two explanations for this data-efficiency gap. First, early in training, language models learn shallow, local patterns before more complex structures [26, 13]. C4 and FineWeb-Edu may therefore devote more capacity to surface regularities than to broadly transferable long-range dependencies and in-context learning. Each NCA sequence, by contrast, requires the model to infer and consistently apply a latent rule. Second, NCA contributes diversity orthogonal to more text: natural language contains substantial redundancy in patterns and topics [1], whereas each NCA sequence represents a distinct function, increasing diversity per token.

Beyond one-size-fits-all pre-training Our complexity ablations reveal a nuanced picture that the optimal distribution for training varies by downstream domain. In Figure 4b, we observed that code benefits from lower-complexity NCA rules, while web text and math benefit from higher-complexity ones, suggesting these domains encode computations of measurably different character. This opens a new axis of control. Rather than treating training data as fixed, we can tune the structures of synthetic data to match the target domain. Unlike grammar-based synthetic tasks, where each formal grammar defines a task with fixed structural complexity, NCAs provide a continuous and tunable spectrum of complexity within a single generator family. If researchers can craft distributions that embody the primitives a domain requires (e.g., rigid state-tracking for code [47], richer long-range dependencies for genomic sequences [82]), they can instill these capabilities directly, without scaling to trillions of general-purpose tokens. The result could aid the development of specialized, small language models [7] that are more efficient to train and deploy—trained not on more data, but on better-matched data.

Limitations and open problems We have demonstrated the effectiveness of NCA pre-pre-training up to 3B parameters and slightly beyond Chinchilla-optimal token budgets (98B tokens), and find that the gains in convergence speed and language modeling persist with model and data scale. Extending these experiments to the multi-trillion token regime of frontier LMs would be a valuable direction for future work. A related question is whether NCA data can serve not just as a pre-pre-training signal but as a scalable substitute for natural language pre-training. For larger alphabet sizes ($n = 10, 15$), we observe a reverse U-shaped trend: downstream improvement peaks at an intermediate token budget and plateaus beyond it—reinforcing that effective synthetic pre-training depends critically on structural choices in the generator, not merely on scale. This points to a key open problem: developing principled methods to guide synthetic generators toward structures that match target domains. Our complexity results show that such matching matters, but gzip compressibility and alphabet size are only two lenses on a multifaceted property. Characterizing which axes of complexity matter for which domains could unlock fully synthetic pre-training at scale. NCA is one point in the vast space of possible synthetic generators; the key insight is not that NCA specifically is optimal, but that structured synthetic data with appropriate complexity can provide meaningful pre-training signal without any linguistic content. The question is no longer whether synthetic pre-training can work, but how to design distributions that maximize what models learn.

Author Contributions

Dan Lee co-lead the project and contributed to all aspects of experiments and writing.

Seungwook Han co-lead the project and contributed to all aspects of experiments and writing.

Akarsh Kumar supported this project, contributed to the design of the experiments and significantly to the writing.

Pulkit Agrawal advised the development of the project idea from inception and contributed significantly to the writing.

Acknowledgment

We want to express our gratitude to Zachary Shinnick, Phillip Isola, Yoon Kim, Ryan Bahlous-Boldi, Idan Shenfeld, Nitish Dashora, Jyo Pari, Aldo Pareja, and members of the Improbable AI lab for the helpful discussion on the paper. The research was supported in part by NSF CSGrad4US Fellowship and Google. The research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-23-1-0277 and W911NF-21-1-0328. Also, the research was sponsored by

the Department of the Navy, Office of Naval Research under Grant Number N00014-22-1-2740. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office, Naval Research Office, or the U.S. Government.

References

[1] Abbas, A., Tirumala, K., Simig, D., Ganguli, S., and Morcos, A. S. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023. URL https://arxiv.org/abs/2303.09540.

[2] An, J., Huang, D., Lin, C., and Tai, M. Measuring gender and racial biases in large language models, 2024. URL https://arxiv.org/abs/2403.15281.

[3] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] Austin, J., Odena, A., Nye, M. I., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., and Sutton, C. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

[5] Baradad, M., Chen, R., Wulff, J., Wang, T., Feris, R., Torralba, A., and Isola, P. Procedural image programs for representation learning. *Advances in Neural Information Processing Systems*, 35: 6450–6462, 2022.

[6] Baradad Jurjo, M., Wulff, J., Wang, T., Isola, P., and Torralba, A. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

[7] Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., and Molchanov, P. Small language models are the future of agentic ai, 2025. URL https://arxiv.org/abs/2506.02153.

[8] Berkovich, J. A. and Buehler, M. J. Lifegpt: Topology-agnostic generative pretrained transformer model for cellular automata. *npj Artificial Intelligence*, 1(1):23, 2025.

[9] Berkovich, J. A., David, N. S., and Buehler, M. J. Automatagpt: Forecasting and ruleset inference for two-dimensional cellular automata. *arXiv preprint arXiv:2506.17333*, 2025.

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

[11] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[12] Chan, S. C. Y., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers, 2022. URL https://arxiv.org/abs/2205.05055.

[13] Chen, A., Shwartz-Ziv, R., Cho, K., Leavitt, M. L., and Saphra, N. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. *arXiv preprint arXiv:2309.07311*, 2023.

[14] Chen, M. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[15] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

[16] Cheng, Z., Cao, M., Pishdad, L., Cao, Y., and Cheung, J. C. Can llms reason abstractly over math word problems without cot? disentangling abstract formulation from arithmetic computation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14317–14344, 2025.

[17] Chiang, C.-H. and Lee, H. Pre-training a language model without human language, 2020. URL https://arxiv.org/abs/2012.11995.

[18] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[19] Cook, J., Sapora, S., Ahmadian, A., Khan, A., Rocktaschel, T., Foerster, J., and Ruis, L. Programming by backprop: Llms acquire reusable algorithmic abstractions during code training. *arXiv preprint arXiv:2506.18777*, 2025.

[20] Copeland, B. J. The church-turing thesis. In Zalta, E. (ed.), *Stanford Encyclopedia of Philosophy*. Stanford Encyclopedia of Philosophy, 2012.

[21] Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[22] Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., Hutter, M., and Veness, J. Language modeling is compression, 2024. URL https://arxiv.org/abs/2309.10668.

[23] Desai, A., Gulwani, S., Hingorani, V., Jain, N., Karkare, A., Marron, M., R, S., and Roy, S. Program synthesis using natural language, 2015. URL https://arxiv.org/abs/1509.00413.

[24] Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

[25] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[26] Evanson, L., Lakretz, Y., and King, J.-R. Language acquisition: do children and language models follow similar learning stages? *arXiv preprint arXiv:2306.03586*, 2023.

[27] Field, D. J. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394, 1987.

[28] Finzi, M., Qiu, S., Jiang, Y., Izmailov, P., Kolter, J. Z., and Wilson, A. G. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *arXiv preprint arXiv:2601.03220*, 2026.

[29] Gardner, M. Mathematical games. *Scientific american*, 222(6):132–140, 1970.

[30] Garg, S., Tsipras, D., Liang, P., and Valiant, G. What can transformers learn in-context? a case study of simple function classes, 2023. URL https://arxiv.org/abs/2208.01066.

[31] Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[32] Gokaslan, A., Cohen, V., Pavlick, E., and Tellex, S. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[33] Han, S., Pari, J., Gershman, S. J., and Agrawal, P. General intelligence requires reward-based pretraining, 2025. URL https://arxiv.org/abs/2502.19402.

[34] Han, S., Song, J., Gore, J., and Agrawal, P. Emergence and effectiveness of task vectors in in-context learning: An encoder decoder perspective, 2025. URL https://arxiv.org/abs/2412.12276.

[35] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

[36] Hu, M. Y., Petty, J., Shi, C., Merrill, W., and Linzen, T. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases. *arXiv preprint arXiv:2502.19249*, 2025.

[37] Hu, M. Y., Petty, J., Shi, C., Merrill, W., and Linzen, T. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases, 2025. URL https://arxiv.org/abs/2502.19249.

[38] Huh, M., Cheung, B., Wang, T., and Isola, P. The platonic representation hypothesis, 2024. URL https://arxiv.org/abs/2405.07987.

[39] Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[40] Jelassi, S., Mohri, C., Brandfonbrener, D., Gu, A., Vyas, N., Anand, N., Alvarez-Melis, D., Li, Y., Kakade, S. M., and Malach, E. Mixture of parrots: Experts improve memorization more than reasoning, 2025. URL https://arxiv.org/abs/2410.19034.

[41] Jiang, H. A latent space theory for emergent abilities in large language models. *arXiv preprint arXiv:2304.09960*, 2023.

[42] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[43] Kataoka, H., Okayasu, K., Matsumoto, A., Yamagata, E., Yamada, R., Inoue, N., Nakamura, A., and Satoh, Y. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[44] Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

[45] Kolmogorov, A. N. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376, 1963.

[46] Li, J., Hou, Y., Sachan, M., and Cotterell, R. What do language models learn in context? the structured task hypothesis, 2024. URL https://arxiv.org/abs/2406.04216.

[47] Li, J., Guo, D., Yang, D., Xu, R., Wu, Y., and He, J. Codei/o: Condensing reasoning patterns via code input-output prediction, 2025. URL https://arxiv.org/abs/2502.07316.

[48] Li, M. and Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, Cham, 4 edition, 2019. ISBN 978-3-030-11297-4. doi: 10.1007/978-3-030-11298-1. URL https://doi.org/10.1007/978-3-030-11298-1.

[49] Li, Y., Bubeck, S., Eldan, R., Giorno, A. D., Gunasekar, S., and Lee, Y. T. Textbooks are all you need ii: phi-1.5 technical report, 2023. URL https://arxiv.org/abs/2309.05463.

[50] Lloyd, S. Measures of complexity: a nonexhaustive list. *IEEE Control Systems Magazine*, 21 (4):7–8, 2001.

[51] Lozhkov, A., Ben Allal, L., von Werra, L., and Wolf, T. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

[52] Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Frozen pretrained transformers as universal computation engines. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 7628–7636, 2022.

[53] Lu, Z., Zhou, A., Ren, H., Wang, K., Shi, W., Pan, J., Zhan, M., and Li, H. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms, 2024. URL https://arxiv.org/abs/2402.16352.

[54] Mirchandani, S., Xia, F., Florence, P., Ichter, B., Driess, D., Arenas, M. G., Rao, K., Sadigh, D., and Zeng, A. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

[55] Mitchell, M. *Complexity: A guided tour*. Oxford University Press, 2009.

[56] Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. https://distill.pub/2020/growing-ca.

[57] Mota, F., Aaronson, S., Antunes, L., and Souto, A. Sophistication as randomness deficiency. In *Descriptional Complexity of Formal Systems: 15th International Workshop, DCFS 2013, London, ON, Canada, July 22-25, 2013. Proceedings 15*, pp. 172–181. Springer, 2013.

[58] Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of gpt-4, 2023. URL https://arxiv.org/abs/2306.02707.

[59] Nadăş, M., Dioşan, L., and Tomescu, A. Synthetic data generation using large language models: Advances in text and code. *IEEE Access*, 13:134615–134633, 2025. ISSN 2169-3536. doi: 10.1109/access.2025.3589503. URL http://dx.doi.org/10.1109/ACCESS.2025.3589503.

[60] Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[61] Papadimitriou, I. and Jurafsky, D. Injecting structural hints: Using language models to study inductive biases in language learning. *arXiv preprint arXiv:2304.13060*, 2023.

[62] Paster, K., Santos, M. D., Azerbayev, Z., and Ba, J. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

[63] Polyanskiy, Y. and Wu, Y. *Information Theory: From Coding to Learning*. Cambridge University Press, 2025.

[64] Reid, M., Yamada, Y., and Gu, S. S. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

[65] Rendell, P. *Turing Universality of the Game of Life*, pp. 513–539. Springer London, London, 2002. ISBN 978-1-4471-0129-1. doi: 10.1007/978-1-4471-0129-1_18. URL https://doi.org/10.1007/978-1-4471-0129-1_18.

[66] Ribeiro, L. C., Bernardes, A. T., and Mello, H. On the fractal patterns of language structures. *PLOS ONE*, 18(5):1–20, 05 2023. doi: 10.1371/journal.pone.0285630. URL https://doi.org/10.1371/journal.pone.0285630.

[67] Ruis, L., Mozes, M., Bae, J., Kamalakara, S. R., Talupuru, D., Locatelli, A., Kirk, R., Rocktäschel, T., Grefenstette, E., and Bartolo, M. Procedural knowledge in pretraining drives reasoning in large language models. *arXiv preprint arXiv:2411.12580*, 2024.

[68] Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. Analysing mathematical reasoning abilities of neural models, 2019. URL https://arxiv.org/abs/1904.01557.

[69] Sharma, A., Czégel, D., Lachmann, M., Kempes, C. P., Walker, S. I., and Cronin, L. Assembly theory explains and quantifies selection and evolution. *Nature*, 622(7982):321–328, 2023.

[70] Shinnick, Z., Jiang, L., Saratchandran, H., Hengel, A. v. d., and Teney, D. Transformers pretrained on procedural data contain modular structures for algorithmic reasoning. *arXiv preprint arXiv:2505.22308*, 2025.

[71] Shinnick, Z., Jiang, L., Saratchandran, H., Teney, D., and van den Hengel, A. Can you learn to see without images? procedural warm-up for vision transformers, 2025. URL https://arxiv.org/abs/2511.13945.

[72] Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

[73] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[74] Trockman, A. and Kolter, J. Z. Mimetic initialization of self-attention layers, 2023. URL https://arxiv.org/abs/2305.09828.

[75] Tunstall, L., Von Werra, L., and Wolf, T. *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[76] Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., and Hobbhahn, M. Will we run out of data? limits of llm scaling based on human-generated data. *arXiv preprint arXiv:2211.04325*, 2022.

[77] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

[78] Wei, J. and Zou, K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks, 2019. URL https://arxiv.org/abs/1901.11196.

[79] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

[80] Wolfram, S. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):1–35, 1984. ISSN 0167-2789. doi: https://doi.org/10.1016/0167-2789(84)90245-8. URL https://www.sciencedirect.com/science/article/pii/0167278984902458.

[81] Wolfram, S. and Gad-el Hak, M. A new kind of science. *Appl. Mech. Rev.*, 56(2):B18–B19, 2003.

[82] Wu, W., Li, Q., Li, M., Fu, K., Feng, F., Ye, J., Xiong, H., and Wang, Z. Generator: A long-context generative genomic foundation model, 2025. URL https://arxiv.org/abs/2502.07272.

[83] Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference, 2022. URL https://arxiv.org/abs/2111.02080.

[84] Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., Lin, Q., and Jiang, D. Wizardlm: Empowering large pre-trained language models to follow complex instructions, 2025. URL https://arxiv.org/abs/2304.12244.

[85] Zhang, S., Patel, A., Rizvi, S. A., Liu, N., He, S., Karbasi, A., Zappala, E., and van Dijk, D. Intelligence at the edge of chaos. *arXiv preprint arXiv:2410.02536*, 2024.

[86] Zhao, R., Saphra, N., and Kakade, S. M. Distributional scaling laws for emergent capabilities. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024.

[87] Zipf, G. K. *Human behavior and the principle of least effort.* Addison-Wesley Press, 1949.

[88] Ziv, J. and Lempel, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. doi: 10.1109/TIT.1977.1055714.

A NCA Rule Generation and Tokenization Pseudo-code

We describe how the NCA rules are sampled according gzip compression ratio and are tokenized for training in the following pseudo-code in Algorithm 1.

Algorithm 1 NCA Rule Generation and Tokenization

Require: Grid size $H \times W$, alphabet size n, patch size p, trajectory length T, temperature τ, complexity threshold $r_{\min}$
Ensure: Rule parameters θ, trajectory $\mathcal{X}$, token sequence S

function SAMPLECOMPLEXNCA($H, W, n, p, T, \tau, r_{\min}$)
  repeat
    $\theta \leftarrow$ INITNETWORK() ▷ random init, frozen for the rollout
    $c^{(0)} \sim \mathcal{U}\{0, \dots, n-1\}^{H \times W}$ ▷ random initial grid
    for $t = 1$ **to** T **do**
      for each cell i **do**
        $c_i^{(t)} \sim \mathrm{softmax}\big(f_\theta(N(c^{(t-1)}, i))/\tau\big)$
      end for
    end for
    $\mathcal{C} \leftarrow (c^{(0)}, c^{(1)}, \dots, c^{(T)})$
    $S \leftarrow$ TOKENIZE($\mathcal{C}, n, p$)
    $r \leftarrow |\mathrm{gzip}(S)| \,/\, |S|$ ▷ compression ratio in $(0, 1]$
  until $r > r_{\min}$
  return $(\theta, \mathcal{C}, S)$
end function

function TOKENIZE($\mathcal{C}, n, p$)
  $S \leftarrow$ empty sequence
  for all grids $c^{(t)} \in \mathcal{C}$ **do**
    Partition $c^{(t)}$ into non-overlapping $p \times p$ patches $\{p_{t,1}, \dots, p_{t,M}\}$ in row-major order, where $M = HW/p^2$
    for all patches $p_{t,j}$ **do**
      token $\leftarrow \sum_{\ell=0}^{p^2-1} p_{t,j}[\ell] \cdot n^\ell$ ▷ base-n numeral over p^2 cells
      Append token to S
    end for
  end for
  return S
end function

B Analysis on Natural and Synthetic Data Distributions

In this section, we examine the distributions of natural-language and NCA-generated synthetic data with respect to two primary high-order heuristics: (1) token frequency distribution and (2) gzip compressibility.

B.1 NCA data exhibits similar token distributions to natural language

To compare the token frequency across different data distributions, we sample and tokenize random text sequences from the natural language datasets (OpenWebText, OpenWebMath, and CodeParrot) and NCA generated data ($n = 15$). Figure 7 shows the distribution of relative token frequencies. Data generated from the NCAs follows a heavy-tailed, Zipfian token distribution that is structured similarly to natural language. Another interesting observation is that natural language depending on the domain varies quite drastically from 32% in code and 60-70% in math and web text.

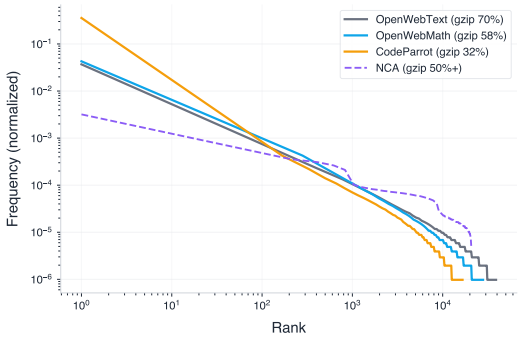


Figure 7: **NCA data exhibits a similar Zipfian or power-law structure to natural language.** We compare the relative token frequency distribution for each of the natural language corpora and NCA data. Natural language from different domains has different average complexity as measured by gzip compressibility (see legend).

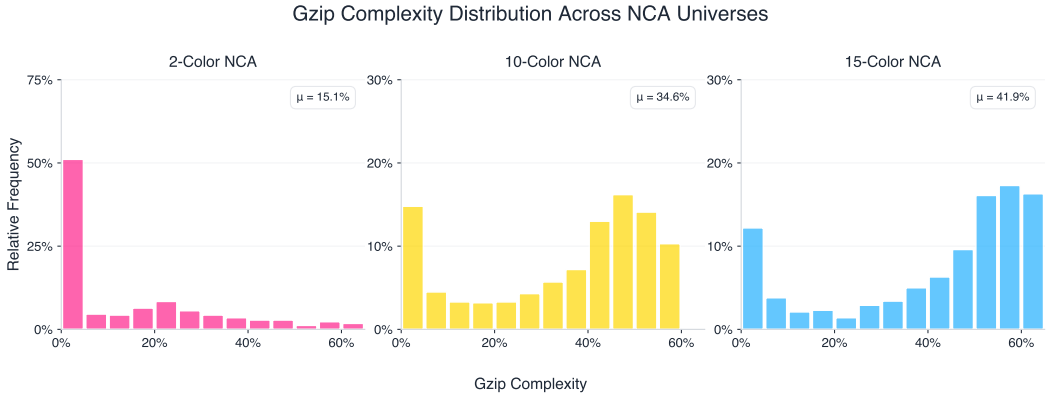


Figure 8: Different NCA alphabet sizes, $n = 2, 10, 15$ naturally yield different complexity distributions of the data. Increasing n inherently increases the complexity of the data.

B.2 Increasing the alphabet size n leads to more complex generated trajectories

Figure 8 compares the distribution of gzip complexity across trajectories generated by randomly sampled NCAs across different alphabet sizes $n = 2, 10, 15$. As n increases, the distribution skews towards less compressible, more complex data. This implies that with higher n, the universe of rules expands and naturally the dynamics become more complex.

C Detailed Pre-pre-training and Pre-training Setup

For our 1.6B parameter Llama-based transformer, we use 24 layers, 32 heads, and hidden dimension of 2048. For the 3B model, we use 26 layers, 32 heads, hidden dimension of 3072.

Table 1 summarizes the hyperparameters used for both pre-pre-training on NCA data and subsequent pretraining on natural language datasets. We sweep various batch sizes (32 to 512), learning rates (1×10^{-3} to 1×10^{-5}), and weight decays (1×10^{-1} to 1×10^{-6}). To ensure reproducibility, we train our pipeline on 4 randomness seeds for each main pipeline (NCA Pre-pre-training, Scratch, and C4 Pre-pre-training) and at least 2 seeds for each ablation run.

We train all models on several nodes of H200, encompassing 20,000+ compute hours across main results and ablations. NCA pre-pretraining data generation cost is negligible, as the generator model is parameterized by a small network that can be run on a CPU to generate the full dataset within minutes.

Hyperparameter	Pre-pre-training	Pre-training	
		1.6B	3B
Effective batch size	32	512	2048
Sequence length	1024 (1.6B) or 2048 (3B) tokens	1024 tokens	2048 tokens
Learning rate	1×10^{-4}	5×10^{-4} (Math / OpenWebText / Fineweb-Edu), 2×10^{-4} (Code)	1×10^{-4} (Fineweb-Edu-3B) 3×10^{-4} (OpenWebText)
LR schedule	Cosine w/ warmup	Cosine w/ warmup	Cosine w/ warmup
Warmup steps (% total)	10%	10%	10%
Optimizer	AdamW	AdamW	AdamW
Weight decay	None	1×10^{-4}	1×10^{-1}
Gradient clipping	None	1.0	1.0

Table 1: Hyperparameters for pre-pre-training and pre-training experiments.

D Detailed Fine-Tuning Setup

For GSM8K and HumanEval, we evaluate on all tasks provided by the benchmarks. For BigBench-Lite, given the quantity and imbalance of samples and tasks, only tasks with above 100 samples were chosen. The exact BBL task list was: bbq_lite_json, formal_fallacies_syllogisms_negation, logic_grid_puzzle, logical_deduction, operators, play_dialog_same_or_different, strange_stories, strategyqa, symbol_interpretation, vitaminc_fact_verification, and winowhy

For GSM8K and Big Bench-Lite, we fine-tune the OpenWebMath and OpenWebText pre-trained models on the respective training sets for 10 epochs using a LR of 1e-5 to enable the models to follow the question answering format for evaluation. For GSM8k, we also fine-tune on the Chain-of-Thought reasoning trace provided by the dataset. FFor HumanEval, we do not fine-tune the models since it is a code completion task. For reproducibility, we train 4 seeds for each model and baseline and report the averages across runs.

We evaluate the models' performances across different Pass@k. For GSM8K and Humaneval, we vary k from 1, 8, 16, and 32. For BigBench, given some tasks are multiple choice, we only report k from 1, 2 and 4, with normalization against random guessing. We use the unbiased estimator from Chen [14], computing the metric from 64 total decodings per run. For evaluation, we sampled HumanEval with a temperature of 0.4 and top-p of 0.95, GSM8K with a temperature of 0.2 and top-p of 0.95 and BigBench-Lite with a temperature of 0.4 and top-p of 0.95. We use 4 training pipeline seeds per task for each baseline and NCA pre-pretrained models and 5 decoding seeds per pipeline seed. For Pass@k evaluation, we sweep generation settings across temperatures of 0.0, 0.2, 0.4, 0.6, and 1.0, and select the best performing single temperature for each task when computing accuracies.

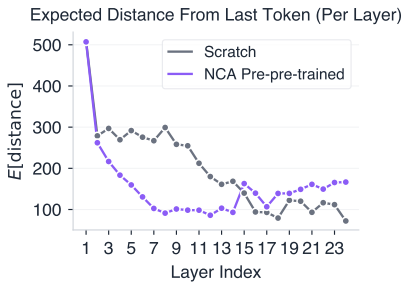


Figure 9: **The expected attention distance from the last token, per layer.** The two models exhibit inverted patterns: the scratch model attends broadly in early layers and narrows with depth, whereas the NCA pre-pre-trained model contracts attention quickly before expanding it again in deeper layers.

E Perplexity and Reasoning Detailed Results and Statistical Significance

In Table 2 we share more detailed tables outlining the exact values and standard deviations for final validation perplexity during pretraining and accuracy at varying pass@k for reasoning benchmarks.

We also compute the statistical significance of reasoning performance of NCA pre-pre-trained language models against natural language and synthetic baselines via Welch's one-tailed T-Test for $n = 4$. As seen in Table 3, though gains across all reasoning benchmarks are modest, NCA pre-pre-training performance on GSM8K is statistically significant compared to all natural language baselines and performance on BigBench is statistically significant compared to all baselines at pass@2 and pass@4. This supports that NCA pre-pre-training can have a modestly positive impact on downstream reasoning performance.

a) Perplexity

Dataset	Scratch	NCA	Dyck	C4	FW-Edu
OpenWebText	14.66	**13.82**	14.36	14.69	14.66
Std Dev	±0.02	±**0.01**	±0.10	±0.04	±0.02
CodeParrot	1.92	**1.84**	1.85	1.89	1.89
Std Dev	±0.04	±**0.00**	±0.01	±0.00	±0.02
OpenWebMath	8.11	**7.70**	7.91	8.14	8.16
Std Dev	±0.03	±**0.02**	±0.07	±0.01	±0.00

b) Reasoning Accuracy (pass@k)

pass@k	Scratch	NCA	Dyck	C4	FW-Edu
GSM8K @1	3.8%	**4.4%**	4.1%	3.8%	3.4%
Std Dev	±0.1%	±**0.3%**	±0.4%	±0.2%	±0.1%
GSM8K @8	17.9%	**19.2%**	18.6%	17.8%	16.8%
Std Dev	±0.3%	±**0.3%**	±0.9%	±0.5%	±0.4%
GSM8K @32	36.6%	**37.9%**	37.4%	36.2%	35.5%
Std Dev	±0.6%	±**0.3%**	±0.7%	±0.9%	±0.4%
HEval @1	6.8%	7.5%	6.9%	6.3%	6.4%
Std Dev	±0.6%	±0.4%	±0.1%	±0.3%	±0.2%
HEval @8	11.2%	11.4%	11.3%	10.5%	10.3%
Std Dev	±0.6%	±0.8%	±0.2%	±0.5%	±0.2%
HEval @32	13.9%	13.8%	**14.3%**	12.6%	12.8%
Std Dev	±0.5%	±1.0%	±**0.4%**	±0.7%	±0.2%
BB-Lite @1	15.4%	15.0%	13.4%	15.9%	12.7%
Std Dev	±1.1%	±1.2%	±2.8%	±0.9%	±1.5%
BB-Lite @2	20.9%	**26.5%**	18.1%	22.8%	23.8%
Std Dev	±2.5%	±**1.0%**	±2.3%	±1.2%	±1.6%
BB-Lite @4	25.9%	**36.5%**	22.7%	29.7%	33.5%
Std Dev	±3.9%	±**2.1%**	±2.1%	±1.3%	±1.7%

Table 2: **NCA pre-pre-training outperforms natural and synthetic baselines across language modeling with modest downstream reasoning gains.** a) Final validation perplexity (lower is better). b) Downstream reasoning performance (higher is better). All baselines use 4 seeds of pre-pretraining and pre-training. **Bold** indicates results statistically significant ($p < 0.05$) compared to the Scratch baseline via a one-tailed Welch's t-test ($n = 4$).

a) Perplexity p-values of NCA vs.

Dataset	vs. Scratch	vs. Dyck	vs. C4	vs. FWE
OpenWebText	**< 0.0001**	**0.0008**	**< 0.0001**	**< 0.0001**
CodeParrot	**0.0132**	**0.0159**	**< 0.0001**	**< 0.0001**
OpenWebMath	**< 0.0001**	**< 0.0001**	**< 0.0001**	**< 0.0001**

b) Reasoning Accuracy p-values of NCA vs.

Task / @k	vs. Scratch	vs. Dyck	vs. C4	vs. FWE
GSM8K @1	**0.0113**	0.1394	**0.0097**	**0.0084**
GSM8K @8	**0.0004**	0.1402	**0.0026**	**0.0011**
GSM8K @32	**0.0074**	0.1292	**0.0134**	**0.0052**
HEval @1	0.0537	**0.0269**	**0.0018**	**0.0122**
HEval @8	0.3520	0.4112	0.0571	0.0841
HEval @32	N/A	N/A	**0.0115**	**0.0314**
BB-Lite @1	N/A	**0.0241**	N/A	**0.0267**
BB-Lite @2	**0.0073**	**0.0012**	**0.0017**	**0.0186**
BB-Lite @4	**0.0031**	**< 0.0001**	**0.0013**	**0.0365**

Table 3: **Significance Tests (One-tailed Welch's t-test, n=4) comparing NCA Pre-pre-training against each baseline.** a) p-values for final validation perplexity improvements. b) p-values for downstream reasoning gains. **Bold** indicates statistically significant results ($p < 0.05$). N/A indicates the comparison baseline outperformed NCA for that specific metric.

F Attention Comparison Between Scratch and NCA Pre-Pretrained Models

We provide more detailed analyses of attention patterns across scratch and NCA Pre-Pretrained models in this section.

To understand how NCA pre-pre-training shapes attention behavior, we measure the *expected attention distance* from the last token at each layer: $\mathbb{E}[\text{distance}] = \sum_{d=0}^{T} d \cdot \text{softmax}(\mathbf{q}^\intercal \mathbf{k}_{t-d})$, where $\mathbf{q}$ is the query vector at position t, $\mathbf{k}_{t-d}$ is the key vector at position $t-d$, and T is the context length. Figure 9 reveals a striking inversion: the scratch model attends globally in the earlier layers and locally in the later layers (15+), while the NCA pre-pre-trained model does the opposite—local first, global later.

In Figure 10, we visualized the expected distance only from the last token, and we extend this analysis to compare the expected distance across the whole sequence length and layers. From these visualizations, we further corroborate that the NCA pre-pre-trained and scratch attention layers indeed learn fundamentally different behaviors and local/global biases of where to attend in the earlier and latter layers of the model.

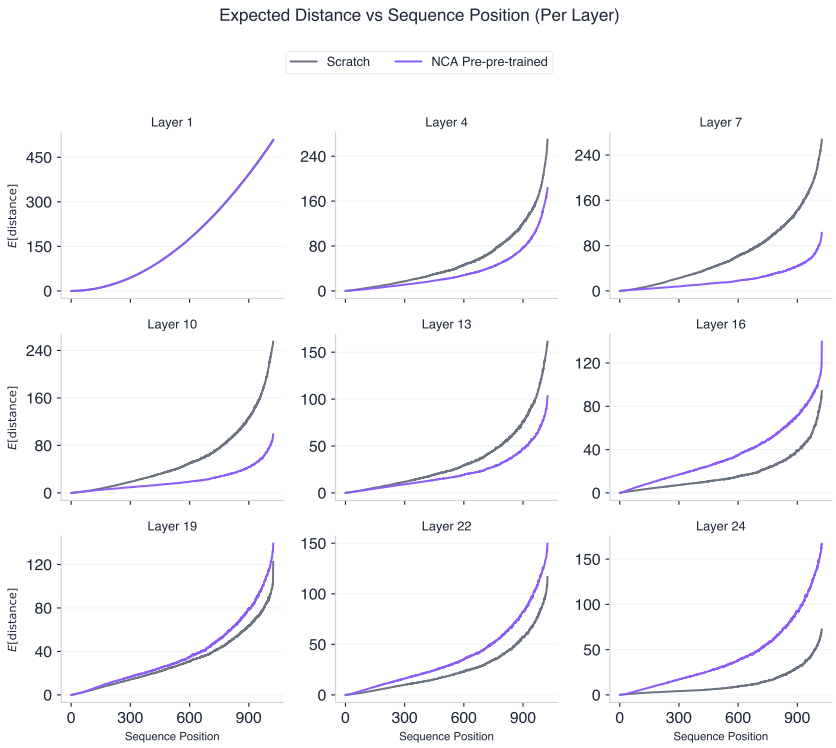


Figure 10: Expected attention distance vs. sequence position, per layer. Each panel shows $\mathbb{E}d = \sum_{j \leq i}(i-j) \cdot \alpha_{ij}$, the average distance from query position i to its attended keys, computed by averaging attention weights across heads. Curves are shown for a model trained from scratch (gray) and one initialized from NCA pre-pre-training (violet). The NCA-initialized model exhibits markedly more local attention patterns across earlier layers and more global patterns in alter layers, suggesting that pre-pre-training induces a different structural bias toward short-range and global dependencies that persists through downstream training.

Table 4: Pass@k performance on MBPP. Results report the mean and standard deviation across runs. Δ denotes the absolute percentage-point improvement of NCA over training from scratch.

Metric	**Temperature**	**Scratch**	**NCA**	Δ
MBPP @1	Greedy	2.45%	**4.30%**	**+1.85**
Std. Dev.	–	$\pm$0.53%	$\pm$0.48%	–
MBPP @1	0.8	1.53%	**2.15%**	**+0.62**
Std. Dev.	–	$\pm$0.10%	$\pm$0.55%	–
MBPP @8	0.8	7.46%	**10.52%**	**+3.05**
Std. Dev.	–	$\pm$0.26%	$\pm$2.42%	–
MBPP @16	0.8	11.51%	**15.57%**	**+4.06**
Std. Dev.	–	$\pm$0.57%	$\pm$3.21%	–
MBPP @32	0.8	16.95%	**21.50%**	**+4.55**
Std. Dev.	–	$\pm$1.33%	$\pm$3.71%	–

G Additional Coding Benchmark: Mostly Basic Python Programs (MBPP)

In this section we provide details on an additional coding benchmark, Mostly Basic Python Programs, used to evaluate Codeparrot pretrained models of 1.6B parameters. As seen in Table 4, NCA models significantly outperform scratch models across various pass@k on the benchmark, albeit absolute performance is low across both models.

H Details on Mechanistic Probe of Induction Heads on NCA Pre-Pretrained Models

In this section, we outline the details of the mechanistic probe analysis conducted on NCA pre-pre-pretrained models. We run this analysis on four independently trained 1.6B NCA models and four architecture-matched random initializations across repeat length $L \in 16, 32, 48, 64, 96$.

For each model and length L, we independently sampled $N = 256$ random sequences $x^{(n)} = (x_1^{(n)}, \ldots, x_L^{(n)})$ from the NCA-trained token vocabulary and constructed the repeated input $r_L^{(n)} = [x^{(n)}; , x^{(n)}]$ together with a control $c_L^{(n)} = [x^{(n)}; , y^{(n)}]$, where $y^{(n)}$ is an independently sampled non-repeating sequence. The two inputs share length and positional structure and differ only in whether the sequence repeats. For a query position q in the second half of $r_L^{(n)}$, the token at q previously occurred at $q - L$, so the standard +1 induction target is $k^\star(q, L) = q - L + 1$: in the pattern $[A][B] \ldots [A]$, a prefix-matching head attends from the final $[A]$ to the earlier $[B]$.

For each head, we score and determine above-null induction heads as follows. Let $[\mathbf{A}^{(m,\ell,h)}(u)]_{q,k}$ denote the post-softmax attention probability from query q to key k for head h in layer ℓ of model m on input u. We computed the content-controlled prefix-matching score

$$s_{m,\ell,h}(L) = \frac{1}{N, |\mathcal{Q}L|} \sum n = 1^N \sum_{q \in \mathcal{Q}_L} \Big(\big[\mathbf{A}^{(m,\ell,h)}(r^{(n)}L)\big]q, k^\star(q, L) - \big[\mathbf{A}^{(m,\ell,h)}(c^{(n)}L)\big]q, k^\star(q, L) \Big)$$

where $\mathcal{Q}_L$ is the set of eligible query positions in the second half; subtracting the control isolates repetition-contingent attention from positional or content-independent biases. At each L, the pooled scores of all heads from the four random-initialized models define an empirical null, and a head is counted as above-null if its score exceeds the null's 99th percentile.

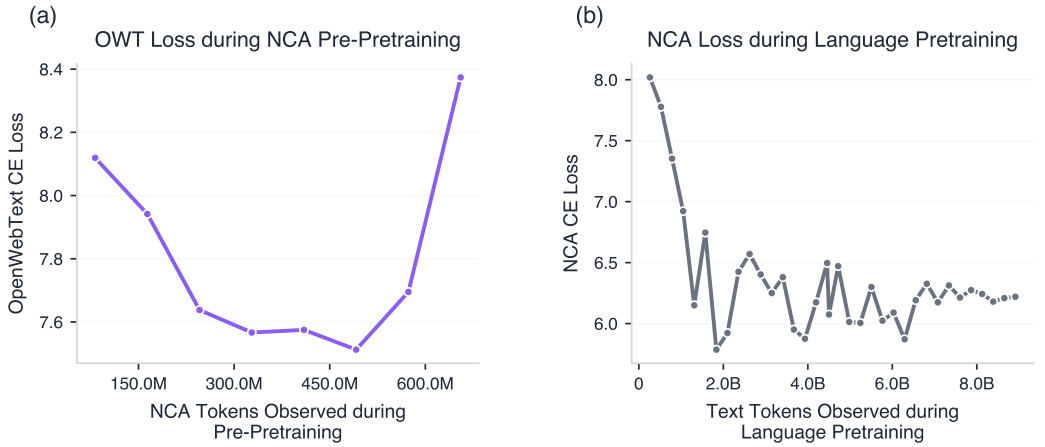


Figure 11: **NCA and OpenWebText are aligned during early stages of pre-pretraining and pre-training, but diverge later on. (a)** Cross-Entropy loss of model during NCA pre-pretraining on OpenWebText. **(b)** Cross-Entropy loss of scratch model during Openwebtext pretraining on NCA data. As seen in both plots, training on NCA and Openwebtext have mild initial positive impact on next token prediction on the other domain.

I Alignment Between NCA and Natural Language

In Figure 11, we investigate the alignment of NCA-generated synthetic data with OpenWebText by analyzing how a model trained on only NCA performs on natural language (Figure 11a), and vice versa (Figure 11b). In both experiments, we only train on one domain (either NCA or language) and do not train on any data or import any weights from the evaluation domain. Since the size of the text tokenizer vocabulary is larger than the NCA vocabulary, for every text data sequence, we map the unique token ids, at most 2048 (max sequence length), to the NCA vocabulary. We find surprising alignment between the two domains; Training on one domain only can lead to improvement in loss

Table 5: **Benefits of NCA Pre-pretraining cannot be recovered from learning-free mimetic initialization.** We report OpenWebText validation perplexity for 400M-parameter models.

Method	**Validation Perplexity** $\downarrow$
Scratch	17.83
NCA	**16.32**
NCA First-Order Statistics-Based Reinitialization	17.77
Mimetic Initialization [74]	17.81

on the other domain. Furthermore, the scaling curve of loss improvement mirrors the curve we observe in Figure 6 for 10-colors. This provides further explanation via alignment for why NCA pre-pretraining may improve language modeling convergence speed and final validation perplexity.

J Comparison to Learning-Free Initializations

In this section, we provide details behind an additional baseline that demonstrates the benefit from NCA pre-pre-training is not just attributable to the statistics underlying the initialization of the pre-training weights. We compare the OpenWebText validation loss of a 400M model initialized using mimetic initialization [74]. Mimetic initialization is a closed form learning-free initialization that biases model weights to generate structural attention patterns observed in pretrained Transformers with the aim of improving optimization and downstream performance. Concretely, it involves decomposing the self-attention query-key and value-output projection vectors as follows: $W_Q W_K^\intercal \approx \alpha_1 Z_1 + \beta_1 I$, $W_V W_{\text{proj}}^\intercal \approx \alpha_2 Z_2 - \beta_2 I$, where Z_1, Z_2 are random normal matrices and $\alpha_1, \alpha_2, \beta_1, \beta_2$ represent hyperparameters for the initialization. As seen in the decompositions, the resulting attention patterns have prominent positive and negative diagonals respectively.

We sweep across various configurations and values for $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \{0, 0.2, 0.5\}$ and report the best performing configuration. The training hyperparameters swept are the same as for the main pre-training sweep. As seen in Table 5, most of the benefit of full NCA synthetic pre-pretraining cannot be recovered from this learning-free initialization, the improvement is marginal at best. We also attempt to try replicating gains by initializing the models with just first-order statistics from NCA pre-pre-trained models and also find it insufficient to cover most of the gains.

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: Abstract and introduction clearly reflect all claims on NCA pre-pretraining, particularly regarding the convergence speed and perplexity improvements, modest reasoning gains reflected in Sections 5 and 6, and how the optimal complexity for the synthetic data distribution varies across domains.

   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: We are clear about the limitations. In the limitations section, given that we have shown that NCA pre-pre-training is beneficial up to the Chinchilla optimal data regime, we discuss how it is important now try to scale further. Moreover, we discuss that our proposed metrics of complexity are only two slices of what it can mean and other metrics should be studied to guide the synthetic data generator even more meaningfully.

   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:

   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: We provide clear guidance on our methodology and experiments to enable full reproducibility. We have provided in Section 3 and 4, the detailed methodology for generating and pre-pretraining language models as well as the detailed information for experimental setup and evaluation. We provided even more details, including the full set of hyperparameters for full reproducibility in Appendices C and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: We will provide the full code and environments to run all experiments after acceptance.

   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: We provide all training and testing details across Sections 3, 4 and Appendices C, D including data setup for pre-pretraining of our method (NCA) and the baselines, the pre-training of language models after pre-pretraining, and the instruction fine-tuning and evaluations of the models.

   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: We provide detailed explanations of error bars and statistical significance across reasoning and perplexity results provided as seen in Appendix E.

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: We provide the details on compute resources in Appendix C

   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: We have reviewed and followed the guidelines provided.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: We discuss the motivations and potential impacts of our work in Sections 6, 1.

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: N/A

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 code, models, and data that are used for pre-pretraining, pre-training and instruction-fine-tuning are cited and publicly available across the work.

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: We have provided detailed information about all datasets utilized throughout the paper as outlined in Section 4.

    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: We do not use any crowdsourcing 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: Our work does not incur risks to study participants (there are none).

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: We use LLMs for writing and editing, but not for non-standard components of this research.

    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.