University of Bonn

CAISA lab

Natural Language Processing Lab

(Summer Semester 2026)

Final Report of Team # 4

Complexity-Guided Pre-Pre-Training

An Evaluation of Synthetic Pattern Transfer in Language Model Pre-Training

Group Members:

Imad Azizi

Rafik Farhane

Instructor: Prof. Dr. Lucie Flek
Advisor: Dr. Nicholas Kluge Corrêa

September 22, 2026

CAISA Lab
https://caisa-lab.github.io/

UNIVERSITÄT BONN

Abstract

Pre-pre-training adds a first training phase on synthetic, rule-generated sequences before a language model ever sees natural text. Published work reports clear gains from this phase and explains them by the complexity of the synthetic data. We tested that explanation at the 670M-parameter scale, with eighteen patterns, four measures of every data source, a twenty-condition transfer study on C4 and a learning-rate sweep on FineWeb-Edu. Pre-pre-training helped, but only two of nineteen conditions beat the baseline by more than a phase-one control containing only trivial constant-token structure does. The best pattern improved C4 perplexity by 2.51 percent and reached the quality of the baseline 1.43× faster. No measure of complexity predicted which pattern helped, and the complexity-matching rule the literature proposes found no support in either direction. Tuning the learning rate of the baseline was worth about six times as much as the best pattern added on top of it.

1. Introduction

Language models rely on a finite supply of web text. A widely cited projection estimates that the stock of public human text could be exhausted between 2026 and 2032 [1]. Rule-generated sequences are inexpensive to generate, do not depend on collecting additional human text, can be produced indefinitely and are fully under experimental control, so synthetic data is one way around that limit. One proposal is *pre-pre-training*. The model is first trained on a synthetic pattern, for example a bracket language, then most of its weights are reset and normal pre-training on natural text follows. The hope is that it picks up something general, for example how to relate one position in a sequence to another, and carries it into the language phase. The extra phase is cheap, so even a small gain would pay for itself. Several studies report such a gain and share one explanation. Transfer is governed by the *complexity* of the synthetic data and is strongest when that complexity is close to that of the target language. That turns the choice of a pattern into a measurement, which is why it is appealing, but it has never been tested on its own and it has a competitor, because a synthetic phase also changes the weights the language phase starts from, and a different starting point interacts with the optimiser whether or not anything transferable was learned.

Our aim was therefore not to repeat the published experiments but to test the explanation behind them. Four questions guided the work. (i) **Does pre-pre-training help at all**, against an otherwise identical baseline that never sees a synthetic phase, and by how much? (ii) **Does complexity predict transfer**, that is, does the compressibility of a pattern predict how well it carries over to natural language? (iii) **If not, which property does?** (iv) **Where does the benefit come from**, an acquired capability or a better starting point for optimisation?

2. Related Work

That non-linguistic sequences can prepare a model for language is older than the two studies motivating this project. Models pre-trained on artificial tokens keep a measurable advantage downstream [2], and music and artificial grammars have been used to probe the inductive biases a transformer brings to language [3, 4]. Transformers have also been trained on cellular automata directly [5], although to reproduce the automaton rather than to ask what the learned representations are worth for language. The strength of this work is control, since a generator exposes the exact rule the model has to find, and its weakness is that the effects are small next to the run-to-run variance of language model training.

Lee et al. [6] train Llama-style models of 400M to 1.6B parameters on rollouts of neural cellular automata selected by gzip compressibility, and report perplexity improvements over a scratch baseline of 8.6 percent at 400M and 5.7 percent at 1.6B with a convergence speedup of 1.4 to 1.6×, all from 164M synthetic tokens. Their targets are OpenWebText, OpenWebMath and CodeParrot, with C4 as a pre-pre-training baseline rather than a target. Their most useful result for our design is an ablation showing that re-initialising the attention weights costs the most transfer, which they read as evidence that attention holds general dependency-tracking machinery. Hu et al. [7] pre-pre-train on formal languages and find k-Shuffle Dyck, the context-sensitive variant in which brackets need not nest, strongest of the four they test, with $k = 64$ bracket types in the main setting and a later sweep up to 256. They reach the loss of their baseline with a 33 percent smaller token budget at one billion parameters and locate an attention subnetwork that survives into the language phase. Their target corpus is C4, as ours is.

The two papers treat the optimiser very differently, and neither reports what that is worth. Lee et al. grid-search learning rate and weight decay per method, so the effects of the data and of the tuning are confounded. Hu et al. instead hold a single peak rate of 5×10^{-4} fixed across all conditions, the rate our own reference recipe inherited, which Section 6.5 shows is far from optimal at this scale. Neither answers how large the choice of pattern is next to the choice of learning rate, the comparison our ablation makes. Hu et al. also include two maximum-entropy controls and report both worse than no pre-pre-training at all, while Lee et al. compare only against scratch, Dyck and C4. Neither includes a minimum-entropy control, the gap our identity condition fills.

Kolmogorov complexity is uncomputable [8], so empirical work replaces it by a compressor, and gzip is the proxy

both papers use, while a second tradition instead measures the information a learner actually extracts, by reading it off the training curve [9, 10]. We use both, and Section 4.3 shows they rank our sources differently. On mechanism, induction heads copy in context by matching an earlier occurrence of the current token and emitting what followed it [11], while the RASP-L characterisation of Zhou et al. [12] separates the functions transformers learn robustly from those needing arbitrary index arithmetic, and Section 6.4 reports a pattern on the wrong side of that line.

3. Methodology

3.1 Design of the study

The study is a controlled comparison. Every model has the same size, sees the same amount of natural text and uses the same optimiser settings. The only thing that changes between conditions is which synthetic data the model saw beforehand, and one condition sees none and serves as the baseline. Any systematic difference at the end therefore comes from the synthetic phase, subject to the yardstick of Section 3.3 and to the phase-one budget confound that Section 5. makes explicit. We also measure every source before training on it, so each carries a number its outcome can be correlated against. The study ran in six steps.

1. Assemble pattern generators spanning a wide range of compressibility and rule-like structure, plus four natural corpora.
2. Measure the four properties of Section 3.2 for every source.
3. Phase one. Train from random initialisation on a single source, under a short token budget matching the reference papers.
4. Reset. Re-initialise every parameter except the attention projections and resize the embedding matrix to the phase-two tokenizer. Lee et al. [6] instead carry all weights but the embeddings across, and their ablation shows attention accounts for most of the transfer, so we keep only attention so any surviving advantage has to be carried by it, and Section 6.6 reports matched runs under their rule.
5. Phase two. Train every condition on the same corpus, token count and schedule. The baseline skips steps 3 and 4 and is otherwise identical.
6. Evaluate on the target corpus validation loss and five benchmarks, and relate the outcome to the profile from step 2 with the tests of Section 3.3.

3.2 Four ways of measuring a data source

Complexity is not one thing, so we describe every source with four quantities. Two are properties of the data alone. The *gzip complexity* g is compressed byte size over original size, so $g \approx 0$ marks highly compressible data and $g \approx 1$ incompressible noise. Values slightly above one occur because the container adds bytes to data DEFLATE cannot shrink. The *oracle loss* is the lowest next-token loss a predictor knowing the generating rule could reach, analytic for the simplest generators and estimated from 10^3 sampled sequences where the rule is drawn afresh per sequence. It is undefined for a natural corpus, so the single condition whose first phase is a corpus enters the oracle-loss correlation with an assumed entropy rate of 1.8 nats per token rather than a measured floor, which is a weakness of that one row.

The other two depend on the data, the model and the compute budget together. Reading the phase-one training curve as a prequential code [9, 10], we split the information content of a token into the part the model learned to predict and the part it did not. The learned part is the epiplexity S_T, the area in bits per token between the training curve and the floor it converges to, and the residual unpredictability at that floor is H_T. They sum to I_T, the total time-bounded information per token, with $\phi = S_T/I_T$ the structural fraction. Compressibility is a property of the data alone, learnability is defined relative to an observer with a fixed architecture and budget, and that difference matters throughout.

3.3 Statistics, and what counts as a real difference

With nineteen pre-pre-trained conditions the sample is small, so the statistics have to be conservative. All associations are Spearman rank correlations, because several measures are bounded or heavy-tailed. Significance comes from permutation tests over 20,000 resamples of the rank vector, and q values control the false discovery rate [13] over one declared family of 62 tests, the eight phase-one measurements against the seven downstream outcomes of Figure 3, plus the six complexity-matching tests of Section 6.3, which re-express the same measurements.

Every condition used a single seed, so run-to-run variance cannot be measured directly and we use two substitutes. For the benchmarks we compare the spread across conditions against twice the binomial sampling standard error of the test set, aggregated over sub-tasks where reported. For the validation loss, where no such error exists, the identity control is the yardstick. A phase one on a single repeated token contains only trivial next-token predictability, so

the improvement it produces bounds the part of any other condition's improvement that can be credited to learning from the data rather than to the warm start itself. That bound is 0.0169 nats.

4. Data Set

The synthetic patterns are what phase one trains on, produced by a rule instead of collected, so we know exactly what there is to learn. The natural corpora are what phase two trains on. We characterise both with the same four measures, so a pattern and a corpus can be compared on one scale.

4.1 The synthetic patterns

The collection holds eighteen generators in five families, chosen to separate compressibility from rule-like structure. The *simple* family holds deterministic rules with almost no compressed size, namely periodic repetition (`periodic`), block copying (`copy`), alternation (`interleaving`), cyclic permutation (`permutation_cycle`), a multi-scale hierarchy (`hierarchical`) and the counting languages $a^n b^n$ and $a^n b^n c^n$. The *Dyck* family holds typed bracket languages with 8, 16, 32 and 64 types, the one rule scaled through four levels of complexity, at the price of a vocabulary growing with k. A closer must match the most recently opened type, so these are the well-nested, context-free k-Dyck languages and not the context-sensitive k-Shuffle Dyck that Hu et al. [7] find strongest. Opening probability is a fair coin with no depth cap, reproducing their harmonic depth distribution. The *NCA* family holds rollouts of neural cellular automata, whose update rule is a small randomly initialised convolutional network mapping the 3×3 toroidal neighbourhood of a cell on a 12×12 grid to a distribution over its ten next states at a fixed temperature. We follow Lee et al. [6] in the essentials, a fresh rule and initial grid per sample, no burn-in, the 2×2 patch tokenisation mapping each block of four cells bijectively to one of 10^4 tokens, and a filter keeping only trajectories above a gzip ratio of 0.5. The one deviation is context length, held at 4,096 tokens so these samples are as long as the others'. `nca_paper` uses the near-deterministic temperature of that study. The two `nca_learnable` conditions raise it so the oracle loss sits at roughly 25 and 50 percent of the uniform baseline. The *composite* family concatenates segments from different generators (`mixer`) or repeats a small palindrome (`composite_mirror_repeat`), and the *control* family holds uniform random tokens and a single repeated token, bounding from both sides what a source without a rule can achieve. Every generator emits exactly 4,096 identifiers and is sampled fresh at every step, so a condition never sees the same sequence twice. Most draw from $[1, 256)$ with 0 as a pad, while the Dyck languages and the automata use vocabularies of $2k + 1$ and 10,003.

4.2 The natural corpora

C4 [14] is filtered but still noisy web text and is the target of the main study. FineWeb-Edu [15] is a quality-filtered educational subset and is the target of the learning-rate ablation. OpenWebMath [16] and CodeParrot [17] are characterised as contrast domains but not trained on. Each was tokenised with the 49,152-token SmolLM2-135M tokenizer [18] and packed into blocks of 4,096 tokens. The C4 shards used in phase two hold 2.54 to 2.61 million blocks, or 10.4 to 10.7 billion tokens, with 8,192 held out for validation. The FineWeb-Edu shard used for the ablation holds 1,367,188 blocks, or 5.60 billion tokens, with 2,048 held out. Phase two consumes 5.24 billion tokens in every case, so no corpus is seen twice. We chose FineWeb-Edu for the ablation because clean data gives a strong baseline and little easy headroom, at the price of a smaller effect size.

4.3 What the data look like

Figure 1 characterises all 22 sources and Table 1 gives the numbers.

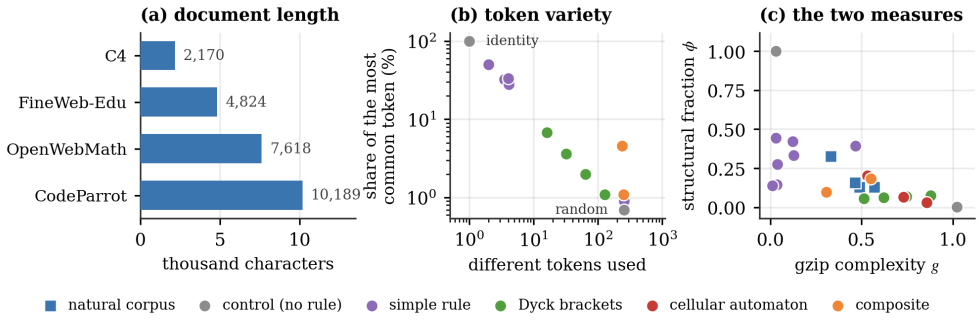


Figure 1: What the data look like. (a) Mean document length of the four corpora, on raw UTF-8 text. (b) Distinct token identifiers in one 4,096-token pattern sample against the share the most frequent one takes, over 200 samples. The automata are off this scale at 10,003 patch tokens. (c) The two complexity measures of Section 3.2, one point per source. Further right is harder to compress, further up means more of it is rule-like structure the model picked up. Table 1 lists every value.

Panel (a) shows that the corpora differ sharply in document length, from 2,170 characters for a C4 document to

Table 1: The measures of Section 3.2 for all 22 sources at the phase-one budget, by family and ordered by g, left block first. g is the mean over 1,000 sequences of 1,024 tokens, S_T and H_T are bits per token, Oracle nats per token. $I_T = S_T + H_T$ is omitted.

Source	Family	g	S_T	H_T	ϕ	Oracle
periodic	simple	0.011	0.580	3.603	0.139	0.006
interleaving	simple	0.030	1.700	2.121	0.445	0.003
hierarchical	simple	0.036	0.798	4.659	0.146	0.004
permutation_cycle	simple	0.038	1.227	3.246	0.274	0.006
counting_anbn	simple	0.121	1.489	2.031	0.423	0.167
counting_anbncn	simple	0.128	1.396	2.782	0.334	0.114
copy	simple	0.465	2.158	3.313	0.394	2.348
dyck_8	dyck	0.514	0.158	2.603	0.057	1.716
dyck_16	dyck	0.622	0.208	3.138	0.062	2.062
dyck_32	dyck	0.746	0.271	3.694	0.068	2.409
dyck_64	dyck	0.879	0.345	4.264	0.075	2.755
nca_learnable_25	nca	0.533	0.996	3.917	0.203	2.853
nca_paper	nca	0.729	0.615	8.965	0.064	0.062
nca_learnable_50	nca	0.856	0.320	9.786	0.032	7.743
mixer	comp.	0.307	0.574	5.277	0.098	0.909
composite_mirror_repeat	comp.	0.552	0.893	3.994	0.183	1.386
identity	control	0.028	0.396	0.000	1.000	0.001
random	control	1.022	0.014	7.994	0.002	5.545
CodeParrot	corpus	0.330	0.579	1.202	0.325	n/a
OpenWebMath	corpus	0.463	0.576	3.047	0.159	n/a
FineWeb-Edu	corpus	0.489	0.583	3.981	0.128	n/a
C4	corpus	0.570	0.625	4.227	0.129	n/a

10,189 for a CodeParrot file, so a fixed block length packs a very different number of documents per block in each. Panel (b) is the surface statistic the patterns were designed around, how many distinct identifiers one 4,096-token sequence uses and what share the most frequent takes. The families separate cleanly over two orders of magnitude, from `identity`, which uses one identifier for every position, through the Dyck languages, which use exactly their $2k$ brackets, to `random`, which uses all 255 with none above 0.7 percent.

The complexity measures behave sensibly on the corpora, the only sources with independent intuitions. Code is the most structured and most compressible of the four, at $\phi = 0.325$ and $g = 0.330$, which fits a domain made largely of rules, while C4 is the least compressible at $g = 0.570$ with a structural fraction indistinguishable from FineWeb-Edu's, 0.129 against 0.128, which fits the idea that what C4 keeps and FineWeb-Edu removes is incompressible without being useful. Three properties matter later. The patterns cover the whole compressibility range, from $g = 0.011$ for periodic repetition to $g = 1.022$ for uniform noise, with the corpora in the middle. The two measures run in opposite directions, panel (c), at $\rho = -0.74$ between g and ϕ, so the sources with more rule-like structure are also the more compressible ones. And identity combines the highest possible structural fraction with the lowest total information of any source, so a high structural fraction does not imply there is much to learn.

5. Experimental Setup

Every value below is fixed across conditions unless an ablation varies it. All experiments use a Llama-style decoder [19] with 28 layers, hidden size 1,536, feed-forward size 3,072, 16 query and 8 key-value heads of dimension 96, RMSNorm, rotary embeddings with $\theta = 50{,}000$, tied embeddings and a context of 4,096 tokens, trained in bfloat16 with FlashAttention [20] and Liger kernels [21] in the Polyglot LLM Foundry code base [22]. Only the embedding matrix differs between phases, so a phase-two model has 670,127,616 parameters against 594.7M to 610.0M in phase one, which uses its own pattern's vocabulary, 256 identifiers for most generators, $2k + 1$ for k-Dyck and 10,003 for the automata, against 49,152 SmolLM2 tokens in phase two. That is also why the H_T column of Table 1 is not comparable across families, since a uniform draw is worth 8.0 bits for a simple pattern but 13.3 for an automaton. Both phases use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1 and gradient clipping at 1.0, under a cosine schedule with 10 percent warmup and seed 1337. Phase one runs at a batch of 16,384 tokens and a peak rate of 1×10^{-4}. Phase two uses a batch of 524,288 tokens for 10,000 steps, which is 5.24B tokens, and a peak rate of 5×10^{-4} unless the ablation varies it. Batch size and learning rate follow DeepSeek-AI [23]. At 7.8 tokens per parameter the phase-two budget sits below the roughly 20 that compute-optimal scaling recommends [24]. We accepted a slightly undertrained baseline to fit twenty conditions into the compute we had, and since every condition shares the budget the comparison between them is unaffected. Phase one ran on one H200, the C4 study on two H200 and the FineWeb-Edu ablation on eight A100 GPUs, about 3,800 GPU hours in total.

The phase-one budget is deliberately short, 2,000 steps, or 32.8M tokens, for fourteen of the eighteen patterns, 7,000 steps or 114.7M for the mixer and 10,000 steps or 163.8M for each automaton condition, the last matching Lee et al. [6]. Equalising it across families would have meant departing from the reference recipes, so we matched them and accepted the confound. It is a real limitation, since phase-one budget is one of the eight predictors in Section 6.3 and reaches $\rho = 0.39$ against the downstream loss on its own, so the effect of a pattern and of how long we trained on it cannot be cleanly separated. The sign is at least not convenient for us, since the conditions given the most phase-one compute transferred worst. Earlier runs at a one-billion-token budget drove every pattern close to its oracle loss and removed the transfer entirely.

The benchmarks are ARC-Easy [25], BLiMP [26], HellaSwag [27], LAMBADA [28] and Winogrande [29], scored zero-shot with the LM Evaluation Harness [30]. The C4 study covers 20 conditions, the 18 patterns plus a corpus-on-corpus null using C4 in both phases and the baseline. The FineWeb-Edu ablation covers eleven patterns and the baseline over seven peak rates from 0.5 to 3.5×10^{-3}, plus six runs in three protocol variants, all on the Bender

cluster of the University of Bonn.[1]

6. Results

6.1 Metrics

The *validation loss* is the mean next-token cross-entropy in nats on the held-out blocks of the target corpus and is the primary outcome. The *perplexity improvement* converts a loss difference against the baseline into a percentage, $1 - \exp(\ell - \ell_{\text{base}})$. The *speedup* is the number of phase-two steps the baseline needs to reach its final smoothed training loss divided by the number the condition needs, so a value above one means the condition got there earlier and one that never reaches it has none. Curves are smoothed with an exponential moving average of span 200 steps wherever compared. Benchmarks are reported as accuracy in percentage points against the baseline, each read against a noise floor of twice the sampling standard error of its test set.

6.2 Downstream transfer on C4

Figure 2 shows what every condition did to the C4 validation loss. Fifteen of the eighteen patterns ended below the baseline and three above it. The best was the mixer, which improved perplexity by 2.51 percent and reached the final quality of the baseline 1.43× faster. The worst was the automaton configuration of the reference paper, the condition following a published setup most closely, at 1.03 percent worse. The whole spread is only 0.036 nats, and read against the identity yardstick of Section 3.3 just two of the nineteen conditions, the mixer and `dyck_16`, beat the baseline by more than a phase-one control containing only trivial constant-token structure does. The extra phase therefore helps on average, but for all but those two by an amount such a warm start already reproduces, which answers question (i).

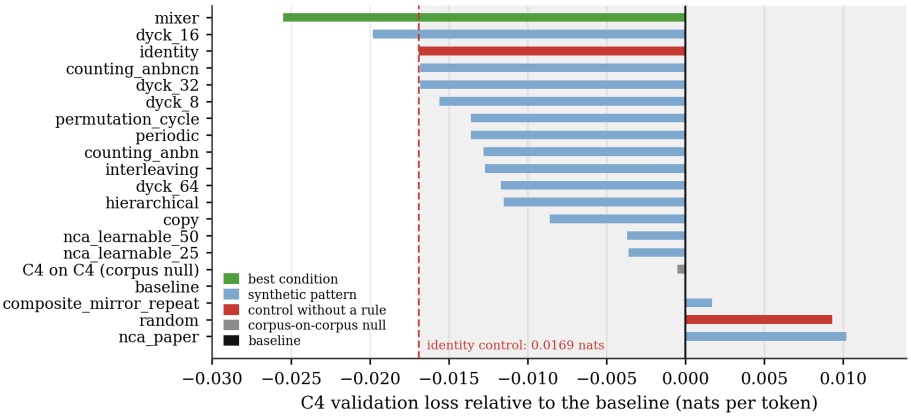


Figure 2: Change in the C4 validation loss for every condition, sorted, negative is better. The grey band marks the 0.0169 nats by which the minimum-entropy identity control beat the baseline. Only two fall outside it on the better side.

Table 2 adds the benchmark scores. The effect is not limited to perplexity, since the strongest patterns improved on almost every benchmark. Two rows are hard to reconcile with a learning account. The identity control gained 1.0 points on BLiMP and 0.6 on HellaSwag although it contains only trivial next-token predictability and no non-trivial dependency structure, and the corpus-on-corpus null, which involves no synthetic data, ended level with the baseline. The spread across the 20 conditions clears the noise floor by a factor of 5.7 on BLiMP, 2.4 on LAMBADA and 1.9 on HellaSwag, barely clears it on Winogrande at 1.4 and does not clear it on ARC-Easy at 0.6, which is why ARC-Easy is left out of the table. We keep it in the correlation matrix as a negative control, because a benchmark carrying no signal should correlate with nothing.

6.3 Complexity as a predictor of transfer

Figure 3 correlates every phase-one measurement with every downstream outcome across the 19 conditions. Gzip complexity reaches $\rho = 0.42$ against the downstream loss, not significant at $p = 0.07$, and absorbed structure S_T is the weakest of the eight at 0.16. The largest value anywhere in the matrix belongs to the total information I_T, at $\rho = 0.65$ with $p = 0.003$, so adding the two components recovers an association neither shows alone. Its sign matters as much as its size. The outcome is a loss, so a positive correlation means the patterns carrying the least information per token transferred best, the opposite of what an account based on richer synthetic data would predict. That answers question (iii). The largest value in any benchmark column is the one cell favouring a complexity account, oracle loss against BLiMP at $\rho = -0.52$ and $p = 0.023$. Two rows are near-identical, because at this budget

[1] Generators, measurement code and the analysis notebooks are in this repository, https://github.com/Nkluge-correa/patterns.

Table 2: C4 transfer for selected conditions, ordered by perplexity improvement. Benchmark columns are changes against the baseline in percentage points. Starred rows are controls. Identity and random hold no non-trivial dependency structure, C4 on C4 repeats the target corpus in both phases. A speedup is undefined when the baseline's final training loss is never reached.

Condition	BLiMP	HellaSwag	LAMBADA	Winogrande	Val loss	PPL impr.	Speedup
baseline, no phase one	n/a	n/a	n/a	n/a	2.9300	n/a	n/a
mixer	+1.0	+1.0	+0.8	+0.6	2.9045	+2.51%	1.43×
dyck_16	+0.5	+0.5	+1.1	+1.6	2.9102	+1.96%	1.40×
identity*	+1.0	+0.6	+0.2	−0.1	2.9131	+1.68%	1.38×
counting_anbncn	+1.3	+0.3	+1.4	+0.6	2.9132	+1.66%	1.38×
eleven further patterns between +1.66% and +0.36%							
C4 on C4*	+0.5	−0.5	−0.7	+0.4	2.9295	+0.05%	1.25×
composite_mirror_repeat	+0.9	−0.0	+0.3	+0.6	2.9317	−0.17%	1.21×
random*	+0.0	−0.4	−0.2	+0.2	2.9393	−0.94%	n/a
nca_paper	+1.0	−0.8	−0.1	−0.9	2.9402	−1.03%	n/a

phase-one H_T and validation loss read the same quantity. ARC-Easy behaves as the negative control it was kept in as, at $|\rho| \leq 0.38$ and $p > 0.11$ everywhere. None of the 62 tests survives correction, the smallest corrected value being $q = 0.10$ and the oracle-BLiMP cell $q = 0.21$, so we report these associations as indicative rather than established.

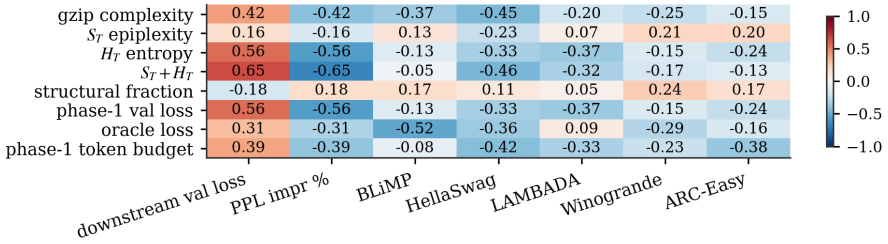


Figure 3: Spearman rank correlations between each phase-one measurement and each downstream outcome over the 19 pre-pre-trained conditions. In the loss column a positive value means a higher measurement went with a worse model.

The stronger form of the hypothesis says transfer is best when the complexity of the pattern is close to that of the target, which Lee et al. [6] state as a practical rule. Correlating $|g_{\text{pattern}} - g_{\text{C4}}|$ with the downstream loss gives $\rho = -0.27$ at $p = 0.26$, neither significant nor in the predicted direction, since a negative sign means patterns further from C4 did slightly better. Replacing g by S_T or H_T weakens it to -0.12 and -0.10, and all six tests land at $q \geq 0.57$. The data support the rule in neither direction rather than contradicting it, so we found no support for complexity matching, which with the weak gzip correlation answers question (ii).

6.4 A limit of compression-based measures

One pair of patterns shows what both measures miss. A forward copy and a mirrored copy are equally determined, since in both the second half of a sequence is fixed by the first, and they share vocabulary, length and token budget. The model learned the forward copy, ending at 2.297 nats against an estimated floor of 2.348. On the mirrored copy it stayed at the uniform ceiling of $\ln 256 = 5.545$ although a perfect predictor could reach 2.773, with S_T at 0.004 bits per token against 0.014 for uniform noise. In a forward copy the token to predict is the one that followed the current token's earlier occurrence, which is exactly what an induction head retrieves [11]. A mirror instead needs a lookback distance that changes at every position, the arbitrary index arithmetic that falls outside RASP-L and that Zhou et al. [12] associate with failure to generalise. Gzip fails on the same input for the same reason, rating the mirror incompressible at $g = 1.00$ against $g = 0.47$ for the copy, because its Lempel-Ziv stage only refers back to earlier forward substrings. Structure the architecture cannot address is therefore indistinguishable from noise under both measures, which is why we dropped the mirrored pattern. Such a null result constrains what is learnable by this observer, not what is present in the data.

6.5 Learning rate against choice of pattern

A possible defence of the complexity account is that complexity governs optimisation stability rather than transfer. Table 3 repeats phase two for eleven patterns and the baseline across seven peak learning rates on FineWeb-Edu. Every condition including the baseline has a pronounced optimum between 1.5 and 2.5×10^{-3}, and moving away from it costs much more than the choice of pattern does at any fixed rate. The Dyck family behaves partly as the complexity account predicts, since its least complex member is the first to lose ground, but the ordering in between is not monotone. Periodic repetition at $g = 0.011$ still trains to within 0.032 nats of its best value at 3×10^{-3} while identity at the almost identical $g = 0.028$ has already diverged to 3.607. Ranking conditions by the highest rate at which they stay within 0.02 nats of their own best value gives a correlation with g of $\rho = +0.39$ at $p = 0.24$, in the predicted direction but not separable from chance. The optima cluster in the same band for almost every condition, which suggests the band belongs to the model scale rather than to the data.

Table 3: FineWeb-Edu validation loss after 10,000 steps against the peak learning rate. The best value per row is in bold, n/a marks a setting we did not run. The bottom three change the protocol. Full transfer keeps all weights, and the last row doubles the warmup and halves the weight decay.

	peak learning rate ($\times 10^{-3}$)						
Condition	0.5	1.0	1.5	2.0	2.5	3.0	3.5
baseline, no phase one	2.7747	2.7049	n/a	n/a	**2.6942**	2.7160	2.7269
dyck_8	2.7612	2.7113	**2.7004**	2.7268	2.7760	n/a	n/a
dyck_16	2.7635	n/a	2.6840	**2.6803**	2.6923	3.0889	n/a
dyck_32	2.7591	2.6973	**2.6831**	2.6920	2.7196	4.6583	n/a
dyck_64	2.7644	2.7017	**2.6853**	2.6938	2.6895	2.6905	2.7399
counting_anbncn	2.7586	2.6925	2.6821	**2.6815**	2.7014	2.7861	n/a
periodic	2.7623	2.6936	**2.6872**	2.6982	2.6923	2.7190	n/a
identity	2.7628	2.6977	**2.6838**	2.7006	2.7788	3.6074	n/a
mixer	2.7528	2.6865	**2.6813**	2.7106	3.0104	n/a	n/a
nca_learnable_25	2.7723	2.7092	2.6958	2.6918	**2.6843**	2.7765	n/a
nca_learnable_50	2.7709	2.7077	2.6933	**2.6898**	2.6950	2.7055	n/a
nca_paper	2.7825	2.7067	**2.6907**	2.6919	2.7122	3.2830	n/a
mixer, full transfer	2.7532	2.6870	**2.6840**	n/a	n/a	n/a	n/a
nca_paper, full transfer	2.7824	n/a	**2.6898**	n/a	n/a	n/a	n/a
mixer, warmup and decay	n/a	n/a	**2.6840**	n/a	n/a	n/a	n/a

The table also quantifies the practical importance of learning-rate tuning relative to pattern choice. Tuning only the learning rate of the baseline, from the 5×10^{-4} of the reference recipe to 2.5×10^{-3}, lowers the validation loss from 2.7747 to 2.6942, which is 7.74 percent of perplexity. The best condition anywhere in the table, `dyck_16` at 2×10^{-3}, reaches 2.6803, a further 0.0139 nats or 1.38 percent. The learning rate is therefore worth about six times as much as the choice of pattern on top of it. Two qualifications apply. The baseline was not run at 1.5 or 2×10^{-3}, so 0.0139 nats is an upper bound on the patterns' contribution, and identity, which contains only trivial constant-token structure, reaches 2.6838 and accounts for about three quarters of that remainder.

6.6 Where the advantage comes from

We measured the advantage over the baseline at eight points in phase two of the C4 study. The synthetic conditions start clearly behind, at -0.101 nats on average at step 200, pull ahead between steps 500 and 2,000, then decay to a small residual. The best peaks at 0.146 nats at step 500 and keeps 0.026 of that by step 10,000, about a fifth. The mean over the eighteen peaks at 0.021 nats at step 2,000 and ends at 0.010. The clearest case is the corpus-on-corpus null, which starts 3.05 nats ahead because its phase one already trained on C4, then gives all of it back. The FineWeb-Edu runs at the matched rate agree, since eight of the eleven patterns lead at step 1,000 by 0.044 nats on average and keep only 0.013 of it by step 10,000. This transient advantage is more consistent with an optimisation or initialisation effect than with a persistent transferred capability, although the present design cannot conclusively rule out the latter, which is our answer to question (iv).

Two protocol checks point the same way. Keeping all weights instead of only the attention projections, the rule of Lee et al. [6], changes the final FineWeb-Edu loss by at most 0.0027 nats across the five matched pairs in Table 3, a fifth of the whole pattern effect and in both directions, and doubling the phase-two warmup while halving the weight decay changes it by the same 0.0027. The structure absorbed during phase two also sits in a narrow band of 0.60 to 0.65 bits per token for all 18 conditions, with no detectable relationship to the structure absorbed during phase one, at $\rho = -0.07$ and $p = 0.77$.

7. Conclusion

Findings. Pre-pre-training produced a small improvement for most patterns, but only two of nineteen conditions beat the baseline by more than a phase-one control containing only trivial constant-token structure. The best improved C4 perplexity by 2.51 percent and converged 1.43× faster, against the 8.6 percent and 1.6× reported at the closest published scale. Neither form of the complexity account was supported, at $\rho = 0.42$ for gzip and $\rho = -0.27$ for matching. The strongest association was the total information per token, at $\rho = 0.65$, whose sign says the sources carrying less information transferred better. The most informative condition is one neither paper includes, a minimum-entropy control that placed third of nineteen. Its phase one converges to a loss of 2×10^{-6}, so it provides only trivial next-token structure, yet it still yields a downstream advantage.

Challenges and limitations. The main difficulty was a mismatch with the reference protocols. Our initial pipeline differed from both papers in the Dyck generator, the weight-transfer rule and, above all, the synthetic token budget, where a thirty-fold larger budget produced no transfer at all, so every result here comes from the corrected pipeline. Two deviations remain, the 4,096-token context of our automaton rollouts and our context-free k-Dyck in place of the k-Shuffle Dyck that Hu et al. [7] find strongest. Three further limits apply. The phase-one budget is not equal across families, the informative regime is the undertrained one, and at roughly 30 GPU hours per phase-two run a single seed was all the compute allowed. Data quality was not a limitation on the synthetic side, where the

generators are exact and we verified each against its rule, but C4 is noisy by construction, which is part of why we repeated the ablation on a filtered corpus.

Interpretation. The practical rule the papers derive, that a pattern should match the target in complexity, was the prediction we could test most directly, and it found no support. The direction of the published effect is reproduced, but its size is roughly a third of the published gain at the closest scale and the mechanism offered for it is not visible in any measurement we made. The time course of the advantage, the size of the learning-rate effect, the behaviour of both controls and the flatness of the phase-two structure point instead towards a better starting point rather than an acquired capability, so on this evidence complexity is not a usable criterion for choosing a pattern.

Future work. Replication with three seeds and further model sizes is the first step, and completing the baseline learning-rate grid would turn our bound on the patterns' contribution into a measurement. Adding k-Shuffle Dyck would test the one published claim our generator does not reach. Our reading suggests a direct test too. A longer plain-text warmup at a fixed step budget should reproduce the benefit without synthetic data. Since no measurement of the data explained which pattern works, inspecting the model is more promising, by comparing the attention circuits each pattern induces and repeating the study with a bidirectional encoder, since the mirror result is specific to causal attention.

References

[1] Pablo Villalobos et al. Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data. *arXiv preprint arXiv:2211.04325*, 2024.

[2] Cheng-Han Chiang and Hung-yi Lee. Pre-Training a Language Model Without Human Language. *arXiv preprint arXiv:2012.11995*, 2020.

[3] Isabel Papadimitriou and Dan Jurafsky. Learning Music Helps You Read: Using Transfer to Study Linguistic Structure in Language Models. *arXiv preprint arXiv:2004.14601*, 2020.

[4] Isabel Papadimitriou and Dan Jurafsky. Injecting Structural Hints: Using Language Models to Study Inductive Biases in Language Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 2023.

[5] Jaime A. Berkovich and Markus J. Buehler. LifeGPT: Topology-Agnostic Generative Pretrained Transformer Model for Cellular Automata. *arXiv preprint arXiv:2409.12182*, 2024.

[6] Dan Lee, Seungwook Han, Akarsh Kumar, and Pulkit Agrawal. Training Language Models via Neural Cellular Automata. *arXiv preprint arXiv:2603.10055*, 2026.

[7] Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between Circuits and Chomsky: Pre-pretraining on Formal Languages Imparts Linguistic Biases. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9691–9709, Vienna, Austria, 2025. Association for Computational Linguistics.

[8] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, fourth edition, 2019.

[9] Léonard Blier and Yann Ollivier. The Description Length of Deep Learning Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[10] Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J. Zico Kolter, and Andrew Gordon Wilson. From Entropy to Epiplexity: Rethinking Information for Computationally Bounded Intelligence. *arXiv preprint arXiv:2601.03220*, 2026.

[11] Catherine Olsson et al. In-context Learning and Induction Heads. *Transformer Circuits Thread*, 2022. arXiv:2209.11895.

[12] Hattie Zhou et al. What Algorithms can Transformers Learn? A Study in Length Generalization. *arXiv preprint arXiv:2310.16028*, 2023.

[13] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300, 1995.

[14] Colin Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[15] Guilherme Penedo et al. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. *arXiv preprint arXiv:2406.17557*, 2024.

[16] Keiran Paster et al. OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text. *arXiv preprint arXiv:2310.06786*, 2023.

[17] Thomas Wolf, Leandro von Werra, and Lewis Tunstall. Codeparrot dataset. https://huggingface.co/datasets/transformersbook/codeparrot, 2021. Accessed: 2026-08-08.

[18] Loubna Ben Allal et al. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. *arXiv preprint arXiv:2502.02737*, 2025.

[19] Hugo Touvron et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

[20] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[21] Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. Liger Kernel: Efficient Triton Kernels for LLM Training. *arXiv preprint arXiv:2410.10989*, 2024.

[22] Polygl0t Project, University of Bonn. LLM Foundry: A cluster-ready codebase for language model development. https://github.com/Polygl0t/llm-foundry, 2026. Accessed: 2026-05-04.

[23] DeepSeek-AI. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[24] Jordan Hoffmann et al. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556*, 2022.

[25] Peter Clark et al. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[26] Alex Warstadt et al. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020.

[27] Rowan Zellers et al. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[28] Denis Paperno et al. The LAMBADA Dataset: Word Prediction Requiring a Broad Discourse Context. *arXiv preprint arXiv:1606.06031*, 2016.

[29] Keisuke Sakaguchi et al. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Communications of the ACM*, 64(9):99–106, 2021.

[30] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, et al. A Framework for Few-Shot Language Model Evaluation. Zenodo, 2023. Version 0.4.0, doi:10.5281/zenodo.10256836.