# PRE-PRE-TRAINING BEFORE PIXELS AND GENES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Foundation models trained on distinct modalities are hypothesized to converge toward similar representations. We turn this observation around and ask whether some universal prior can be learned from synthetic data before exposure to natural data and how the learned priors transfer across different domains. We investigate this question through synthetic pre-pre-training, an initial stage where transformers are first trained on data generated by structured processes followed by standard domain-specific training. We treat synthetic pre-pre-training as a design space, varying the synthetic substrate (neural cellular automata or Dyck formal language), training objective (masked reconstruction or autoregressive prediction), and target domain (vision and genomics). Genomics is a particularly interesting domain since there is abundant sequence data, but limited high-quality annotated data. Across three vision benchmarks and ten genomics tasks, we observe that synthetic pre-pre-training generally improves downstream performance, but that its effectiveness depends on our studied three design factors. In vision, masked reconstruction consistently outperforms autoregressive prediction for both pre-pre-training sources. Masked 2D NCA pre-pre-training achieves the highest accuracy on all three benchmarks, improving upon the randomly initialized and trained baseline by $3.7\%$ on average. In genomics, both synthetic sources transfer effectively, with autoregressive pre-pre-training on Dyck sequences outperforming the randomly initialized baseline by approximately 6 points on NTv3 mean score and 2.6% on the Genomics Classification tasks. These results suggest that synthetic pre-pre-training is a promising way to instill general computational priors before domain-specific pre-training.

## 1 INTRODUCTION

A striking phenomenon in foundation models is that models trained on different modalities converge towards aligned representations despite differing data, architectures, and objectives (Huh et al., 2024; Maniparambil et al., 2024; Li et al., 2024). Such convergence suggests that models share a set of computational structures across modalities: integrating information over long contexts, propagating information via hidden states, and inferring common patterns. If so, these primitives could be learned from synthetic data before training on natural data. We call this stage *pre-pre-training* (Hu et al., 2025a; Lee et al., 2026). In this paper, we ask: **can general computational priors be learned through pre-pre-training and transferred across multiple domains?**

Prior work on synthetic training provides encouraging evidence but only within individual domains. They study the limited effect of each synthetic data source for a corresponding modality. For language, formal languages and neural cellular automata have been used to improve subsequent language training (Lee et al., 2026; Papadimitriou & Jurafsky, 2020; Hu et al., 2025b; Jiang et al., 2026). Separately, symbolic procedural warm-up has been shown to improve vision transformers on image classification tasks (Shinnick et al., 2025). These studies demonstrate that useful structure can be learned without target-domain semantics, but it remains unclear whether synthetic training transfers across qualitatively different domains and what variables govern the transfer.

We address these questions by studying synthetic pre-pre-training across three axes: the *synthetic source*, the *pre-pre-training objective*, and the *downstream target domain*. We compare two sources with complementary structure: Neural cellular automata (NCA) and K-Dyck. NCA dynamics produce long-horizon emergent patterns through repeated application of latent local update rules. With the rule hidden, modeling NCA evolution requires implicit Bayesian inference over the underlying



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

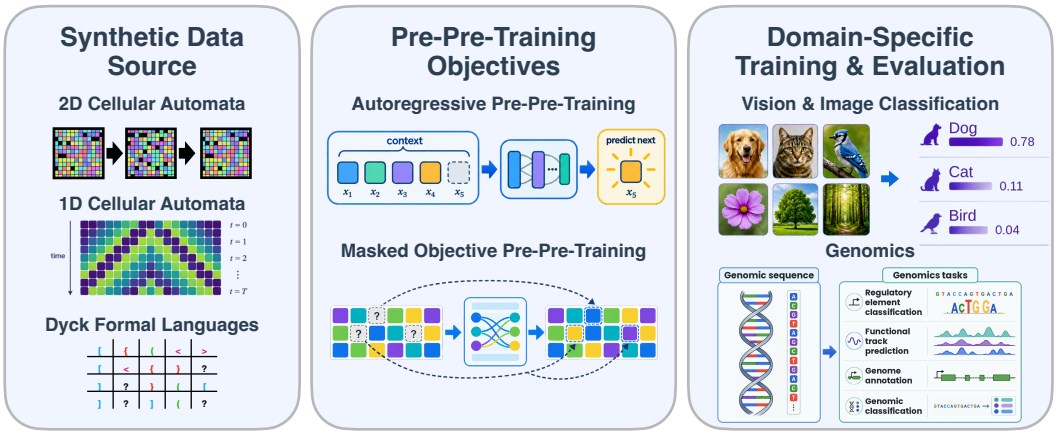



Figure 1: **Overview of Synthetic Pre-Pre-Training.** We pre-pre-train transformer models on synthetic substrates ranging from neural cellular automata to Dyck formal languages, and subsequently train them on vision and genomic data. Transfer performance depends on both the synthetic data formulation and the pre-pre-training objective.

transition dynamics. Dyck languages produce sequences with hierarchical dependencies. A model must implicitly simulate a stack-like memory to track which brackets remain open and in what order. We evaluate transfer to both vision and genomics, two domains with distinct structures and learning challenges. Genomics provides a particularly compelling testbed. Although unlabeled genomic sequences are abundant, generating high-quality functional annotations is challenging (de Almeida et al., 2025). Synthetic pre-pre-training could be especially valuable in this setting by providing transferable computational priors that support learning from limited supervision.

Our results show that synthetic pre-pre-training can benefit models across both vision and genomics. Our contributions are as follows:

1. **A systematic study of the synthetic pre-pre-training design space.** We investigate how synthetic source, training objective, and target domain shape transfer. We compare neural cellular automata and Dyck languages under autoregressive and masked reconstruction objectives in both vision and genomics. Although synthetic pre-pre-training benefits both domains, no single configuration is universally optimal. The best configurations outperform training from random initialization by an average of 3.7% across three vision benchmarks, 6 points in mean NTv3 score across three tasks, and 3.4% across the seven Genomic Classification Tasks.
2. **Masked reconstruction is better suited to vision.** Across both synthetic sources and all vision benchmarks, pre-pre-training with the masked reconstruction objective consistently outperforms the autoregressive objective. Masked 2D NCA pre-pre-training achieves the highest accuracy on every benchmark, resulting in an average improvement of $\sim 3.7\%$ over the randomly initialized baseline.
3. **The effects of synthetic pre-pre-training complement and persist after extensive natural-data training.** Downstream benefits remain evident after $\sim 21$ billion tokens of genomic pre-training and task-specific fine-tuning, demonstrating that synthetic initialization can provide value even after substantial exposure to natural sequences. These benefits could be particularly valuable in annotation-constrained domains like genomics, where unlabeled sequences are abundant, but high-quality functional annotations are difficult to obtain.

## 2 Related Works

**Universal representations.** Independently trained models often converge in representational geometry despite differences in initialization, architecture, and objective, with alignment generally increasing with data, capacity, and task performance (Morcos et al., 2018; Bansal et al., 2021; Roeder et al., 2020). Recent work extends this pattern across modalities: vision and language models exhibit partially aligned semantic neighborhoods, and several studies report stronger alignment for larger or

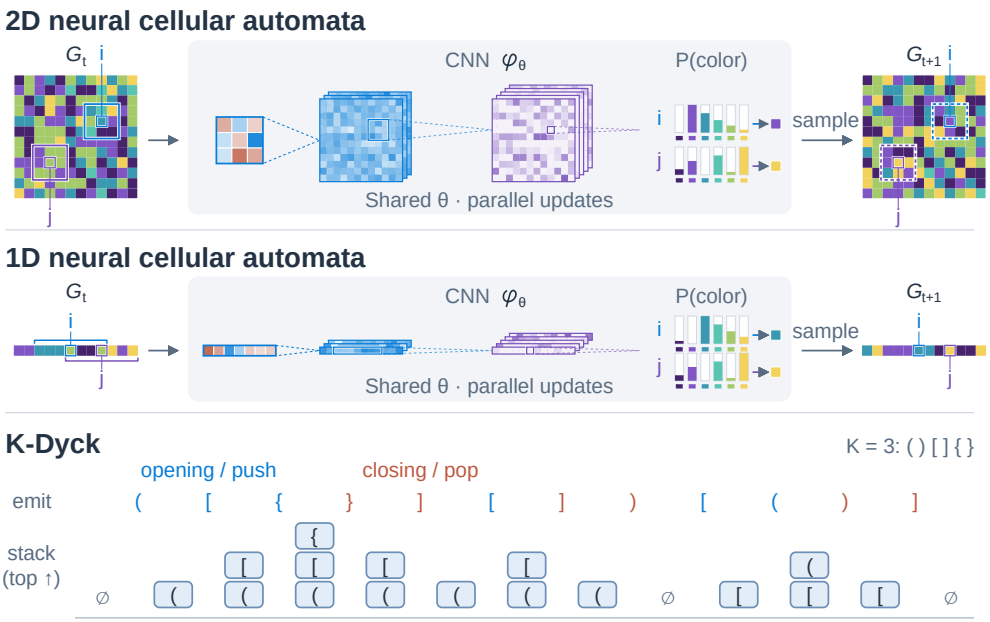



Figure 2: **Synthetic data generation process.** (Top) 1D and 2D Neural cellular automata pre-pre-training data is generated by evolving a grid using an update rule parameterized by the same convolutional neural network. The CNN is applied to all cells in the grid in parallel. This results in a distribution over next state values that are sampled to generate the next grid. (Bottom) K-Dyck formal language pre-pre-training data is generated by simulating a stack memory. The sequence is generated by randomly adding open brackets to the stack or adding closing brackets (stack pop).

more capable models (Huh et al., 2024; Maniparambil et al., 2024; Li et al., 2024). Although this convergence may not be absolute (Tjandrasuwita et al., 2025; Ciernik et al., 2025; Gröger et al., 2026; Koepke et al., 2026), we view this representational convergence as motivation for shared computation.

**Synthetic pre-pre-training and structured initialization.** Structured synthetic data can provide useful inductive biases before a model encounters natural data. Artificial languages and semantics-free reasoning tasks transfer dependency tracking and algorithmic structure to language and mathematics (Papadimitriou & Jurafsky, 2020; Wu et al., 2022; Hu et al., 2025b). More recent procedural warm-ups using formal languages, algorithms, and neural cellular automata improve downstream loss, convergence, and reasoning, with transferable structure appearing in both attention and MLP blocks of transformers (Jiang et al., 2026; Lee et al., 2026). Beyond language, training on formal grammars (Shinnick et al., 2025) and procedurally generated images (Kataoka et al., 2020; Baradad et al., 2022; Baradad Jurjo et al., 2021) have shown to improve visual representations. We extend these single modality studies to a multi-domain study and ablate between different variables of synthetic source, training objective, and downstream domain.

## 3 Method

We pre-pre-train a transformer on trajectories of randomly sampled neural cellular automata or formal language sequences, then transfer its backbone weights into one of the following downstream domains: vision or genomics. We now define the data (Section 3.1, Section 3.2), the two pre-pre-training objectives we use (Section 3.4), and the transfer procedure (Section 3.5). The pre-pre-training data generation process for both synthetic substrates is summarized in Figure 2.

### 3.1 Neural cellular automata

**Cellular automata.** A cellular automaton is a system of cell grids that evolves over time through a shared local update rule. We will denote the grid at time $t$ as $S^{(t)}$. Each cell $i$ holds a state $s_i^{(t)} \in \mathcal{S} = \{0, \ldots, n-1\}$ at time $t$, and the grid has periodic boundaries. An update rule is a

function $\varphi$ that reads a contiguous region of cells centered on $i$ – either a $k \times k$ block (2D) or a $k$-length line (1D) – written as $s^{(t)}_{N(i)}$, and returns that cell's next state: $s^{(t+1)}_i = \varphi\big(s^{(t)}_{N(i)}\big)$. At each iteration, the same $\varphi$ is applied to every neighborhood of cells, so the whole automaton is defined by a single local function and the initial condition of the grid. Although the rule is local and finite, iterating it can produce structures far larger than its neighborhood (Wolfram, 1984). A classic example of these rules are Conway's Game of Life.

**Why these trajectories can be useful pre-pre-training data.** A trajectory $(S^{(1)}, S^{(2)}, \dots)$ represents grids of cells starting at $S^{(1)}$ rolled out over time via a $\varphi$ cellular automata update rule applied to the entire grid. During pre-pre-training, the rule that produced it is never shown. A model that predicts one part of the trajectory from another must infer the rule from the frames in context and then apply it. Since we sample a fresh rule $\varphi$ and initial condition $S^{(1)}$ for every trajectory, the model cannot just memorize a particular rule, but must learn to infer the underlying rule given the context. Since the data carries no meaningful semantics or vocabulary, anything that transfers should be a general computational skill.

**From hand-designed rules to neural rules.** Cellular automata update functions are typically hand-designed lookup tables. Neural cellular automata (NCA) replace the lookup table with a small neural network (in the order of thousands of parameters) $\varphi_\theta$ applied identically at every cell (Mordvintsev et al., 2020). We use this network only as a source of rules: $\theta$ is randomly initialized and *never trained*. Each sampled $\theta$ represents a different automaton, which turns the choice of a rule into a draw from a continuous distribution.

**Two grid geometries.** We instantiate the construction of cellular automata for both 1D ($d = 1$) and 2D ($d = 2$), matching the shape of the downstream data. The 2D automaton lives on a torus which matches the shape of vision data, where the downstream model consumes 2D image patches; the 1D automaton lives on a ring and matches the shape of genomics data, where the downstream model consumes a linear sequence of nucleotides as seen in Figure 1. For both grid geometries, we fix the grid size at $g$ (1D) or $g \times g = 12 \times 12$ (2D). Only the grid geometry is changed: the rule network, the sampling procedure, the complexity filter, and the two training objectives in Section 3.4 are identical in both cases.

**Sampling a trajectory.** Cell states are one-hot encoded, and $\varphi_\theta$ maps a neighborhood to $n$ logits, corresponding to possible cell states. $\varphi$ is a convolutional neural network consisting of the following layers: a $k^d$ convolution with $4$ channels, where $k$ represents the neighborhood size; a $1 \times 1$ convolution with $16$ channels; a ReLU; and a $1 \times 1$ convolution to $n$ outputs. The convolutions are computed with periodic padding, so the rule is local and applied the same everywhere. The cells update in parallel as follows,

$$s^{(t+1)}_i \sim \mathrm{Cat}\Big(\,\mathrm{softmax}\left(\varphi_\theta(s^{(t)}_{N(i)})/\tau\right)\Big), \tag{1}$$

where the temperature $\tau$ is small ($10^{-3}$). To sample an initial grid, we draw one logit vector $\ell \sim \mathcal{N}(0, I_n)$ per rule and then draw each cell independently, $s^{(1)}_i \sim \mathrm{Cat}(\mathrm{softmax}(\ell))$. We roll the automaton forward, discarding the first $t_0$ frames so that we record settled behavior rather than the initial transient dynamics, and keep the next $T$ frames $S^{(t_0+1)}, \dots, S^{(t_0+T)}$, where $S^{(t)} \in \mathcal{S}^{g^d}$.

**Tokenization.** We turn a trajectory into a token sequence patch-wise, as in vision transformers (Dosovitskiy et al., 2021). Each frame is cut into $P = (g/p)^d$ non-overlapping patches of $p^d$ cells: segments of $p$ consecutive cells in 1D, $p \times p$ blocks in 2D. A patch is a vector $u \in \mathcal{S}^{p^d}$, which we read as a base-$n$ number,

$$\iota(u) = \sum_{j=1}^{p^d} u_j \, n^{\,j-1} \in \{0, \dots, n^{p^d} - 1\}, \tag{2}$$

so distinct patches get distinct tokens. Working at the patch level rather than the cell level matches the patch interface of the downstream model for images (Dosovitskiy et al., 2021) and makes each prediction an $n^{p^d}$-way choice instead of an $n$-way one. We serialize the patches of each frame in

scan order (left to right in 1D, row-major in 2D), wrap each frame in the delimiters <grid> and </grid>, written $\delta_{\mathrm{s}}$ and $\delta_{\mathrm{e}}$ below, and concatenate frames until we reach a context length $L$:

$$x = \big( \underbrace{\delta_{\mathrm{s}},\, \iota(u_1^{(1)}),\, \ldots,\, \iota(u_P^{(1)}),\, \delta_{\mathrm{e}}}_{\text{frame 1}},\, \underbrace{\delta_{\mathrm{s}},\, \iota(u_1^{(2)}),\, \ldots,}_{\text{frame 2}}\, \ldots \big) \in \mathcal{V}^L. \tag{3}$$

Each frame therefore contributes $P + 2$ tokens. The vocabulary $\mathcal{V}$ holds the $n^{p^d}$ patch tokens plus the two delimiters and a padding token. Since $n^{p^d}$ grows quickly in $d$, the two settings use different $(n, p)$; values are given in Section 4.

### 3.2 Dyck Formal Languages

**K-Dyck formal languages.** The $K$-Dyck language is a context-free language of balanced brackets with a hierarchical, nested structure (Chomsky & Schützenberger). The parameter $K$ controls the number of distinct bracket pairs. A sequence is valid if all brackets are paired by type and properly nested. For example, `({})[][]` is a valid 3-Dyck sequence.

**Why dyck sequences can be useful pre-pre-training data.** Modeling Dyck sequences requires the transformer to implicitly model a stack-like memory to maintain consistency and produce acceptable bracket types. A model cannot just randomly output a closed bracket as it needs to maintain consistency with the bracket sequence. Like NCA, the data carries no semantic information or meaningful vocabulary, so any artifacts that transfer should result from computation skills acquired from modeling the language.

**Sampling a trajectory.** We generate a Dyck trajectory one bracket at a time while maintaining a stack of unmatched opening brackets. At an opening step, we sample one of the $K$ bracket types uniformly, emit its opening bracket, and add its type to the stack. At a closing step, we emit the closing bracket corresponding to the most recent unmatched opening bracket and remove that type from the stack. Thereby, we generate balanced Dyck sequences with variable length capped at length 1024.

**Tokenization.** We represent each bracket with a single token. The $K$ opening brackets receive token IDs $0, \ldots, K-1$, and their corresponding closing brackets receive IDs $K, \ldots, 2K-1$, giving a vocabulary of $2K$ bracket tokens.

### 3.3 Mimetic Initialization

Mimetic initialization (Trockman & Kolter, 2023) is a closed-form, learning-free initialization of the weights of self-attention layers. The goal is to initialize transformers such that their attention patterns mimic those of pre-trained vision models. It decomposes query–key vectors as $(W_Q W_K^\top \approx \alpha_1 Z_1 + \beta_1 I)$, resulting in a positive diagonal, and the value–output-projection as $(W_V W_{\mathrm{proj}} \approx \alpha_2 Z_2 - \beta_2 I)$, resulting in a negative diagonal. Here $(Z_1, Z_2)$ are randomly sampled Gaussian matrices and $\alpha_1, \alpha_2, \beta_1, \beta_2$ are hyperparameters for the initialization. These target products are then factorized via singular value decomposition (SVD) to obtain the attention weight matrices. In this work, we examine mimetic initialization as a useful baseline to study whether the benefits from synthetic pre-pre-training can be reproduced without training.

### 3.4 Pre-pre-training objectives

We pre-pre-train a transformer $q_\psi$ on these synthetic sequences and use one of two training objectives, described below. Both objectives use the same architecture and data, but differ in the attention mask and loss function.

**Autoregressive prediction.** The first objective is next-token prediction with a causal attention mask:

$$\mathcal{L}_{\mathrm{AR}}(\psi) = -\sum_{j=1}^{L-1} \log q_\psi\big(x_{j+1} \mid x_{\le j}\big), \tag{4}$$

All positions are scored, including delimiters and excluding padding.

**Masked reconstruction.** The second objective masks out randomly selected portions from the generated sequences and trains to reconstructs them with full, non-causal attention. Let $\mathcal{E} \subseteq \{1, \ldots, L\}$ be the eligible positions in the generated sequences. Each eligible position is hidden independently with probability $m$, giving a masked set $\mathcal{M} \subseteq \mathcal{E}$, and the corrupted input replaces those positions with a reserved symbol,

$$\tilde{x}_j = \begin{cases} \texttt{[MASK]} & j \in \mathcal{M} \\ x_j & \text{otherwise,} \end{cases} \qquad \mathcal{L}_{\text{Masked}}(\psi) = -\sum_{j \in \mathcal{M}} \log q_\psi\big(x_j \mid \tilde{x}\big). \tag{5}$$

Only hidden positions are scored. We force $|\mathcal{M}| \geq 1$ so every sequence contributes a loss. The symbol `[MASK]` is a reserved ID outside the patch vocabulary.

### 3.5 Transfer

Pre-pre-training and downstream training use different input representations and prediction targets. We therefore transfer only the shared transformer backbone $\psi_{\text{backbone}}$: the `[CLS]` token (only vision), learned positional embeddings (only vision), transformer blocks, and final layer norm. The patch projection and classification head are randomly initialized for both tasks.

During NCA pre-pre-training, we use rotary positional embeddings for genomics and learned positional embeddings for vision. When transferring the backbone to vision tasks and when the data have different sequence lengths, we resize its learned positional embeddings to match the number of image patches by sampling at evenly spaced locations along the original embedding sequence. At each location, we blend the two neighboring embeddings: $p_{\text{new}} = (1-\alpha)p_{\text{left}} + \alpha p_{\text{right}}$, where $\alpha$ is the fractional distance from the left position to the right. For example, a location halfway between two positions uses their average ($\alpha = 0.5$). The `[CLS]` positional embedding is transferred unchanged.

## 4 Experimental Setup

We evaluate the proposed synthetic pre-pre-training methods in two domains: vision (Section 4.1) and genomics (Section 4.2). In every comparison, we hold the downstream pipeline fixed—data, architecture, optimizer, schedule, and augmentation—and vary only how the transformer backbone is initialized. Thereby, we isolate the contribution of the pre-pre-training on the downstream transfer. We detail our pre-pre-training hyperparameters in Appendix A.

### 4.1 Downstream Evaluation on Vision Tasks

**Model.** We use a ViT-Tiny (Dosovitskiy et al., 2021) model with 12 transformer blocks, 3 attention heads, width 192, learned absolute positional embeddings, bidirectional attention, `[CLS]` pooling, and a linear classifier ($\approx 5.4$M parameters).

**Datasets and Evaluation.** We evaluate on standard image classification benchmarks, CIFAR-100 (Krizhevsky, 2009), TinyImageNet (Le & Yang, 2015), and STL-10 (Coates et al., 2011). STL-10 represents a label-scarce setting: we train only on its 5,000 labeled images and do not use the unlabeled split, mirroring the data-constrained regime that motivates this work. We report the final top-1 accuracy on the test set and standard deviation over three independent training runs. Each run repeats both pre-pre-training and fine-tuning.

**Vision Fine-tuning.** We use a fixed DeiT-style recipe (Touvron et al., 2021) tuned on the randomly initialized baseline for every model: AdamW (Loshchilov & Hutter, 2019) with learning rate $2 \times 10^{-3}$, weight decay $0.05$, cosine learning rate decay, a 50-epoch warmup, 300 training epochs, gradient clipping at $1.0$, and mixed precision, BF16. Augmentations include random cropping, horizontal flipping, RandAugment (2 operations, magnitude 9) (Cubuk et al., 2019), label smoothing of $0.1$, and either mixup ($\alpha = 0.8$) or CutMix (Yun et al., 2019) ($\alpha = 1.0$), selected per batch.

**Pre-Pre-Training Methods.** We compare three families of backbone initialization under the same fine-tuning recipe: (i) *no pre-pre-training*, comprising of random and mimetic initialization (Trockman & Kolter, 2023); (ii) *synthetic-image supervised pre-pre-training* on the 1,000 procedurally

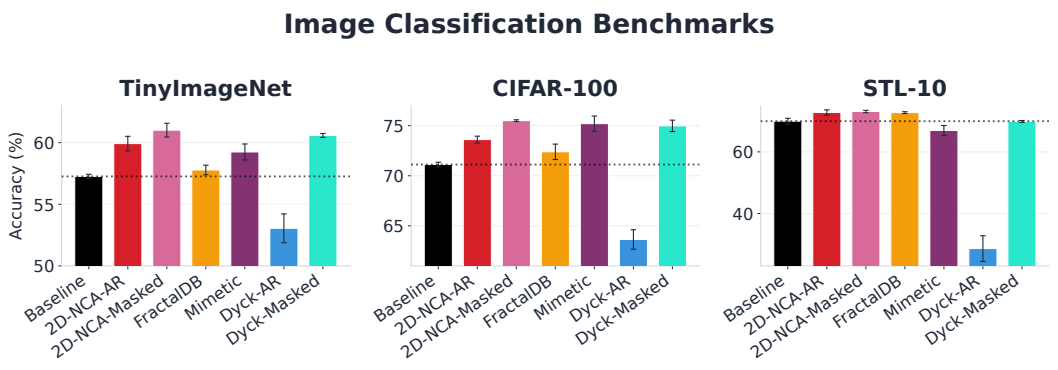



Figure 3: **NCA Pre-pre-training with masked autoencoding outperforms other synthetic and initialization baselines across all image classification tasks.** We report final classification accuracies and standard deviations across six random seeds for TinyImageNet, CIFAR-100, and STL-10.

generated FractalDB classes (Kataoka et al., 2020); and (iii) *sequence pre-pre-training* on NCA data (Lee et al., 2026) or Dyck languages (Shinnick et al., 2025) using either autoregressive modeling or masking objectives.

### 4.2 Downstream Evaluation on Genomics Tasks

**Model.** We use a Llama-based 24-layer transformer model with rotary position embeddings and a final RMS normalization layer. The transferable backbone consists of approximately 400M parameters. For genomic modeling, we use a byte-level tokenizer with a vocabulary of 512 and a context length of 1,024.

**Datasets and Evaluation.** We evaluate on selected tasks from NTv3Bench (Boshar et al., 2025) and Genomics Classification Tasks (Grešová et al., 2023). From NTv3Bench, we select three species–task configurations: human functional-track prediction (34 tracks), chicken functional-track prediction (14 tracks), and human genome annotation (exon, intron, splice acceptor, and start codon). These tasks use 32k-base-pairs for training and the test set provided by the benchmark. We report the mean test-set Pearson correlation across functional tracks and the mean Matthews correlation coefficient (MCC) across annotation tracks.

For the Genomics Classification Tasks, we select seven classification tasks: (1) Human Non-Tata Promoters (2) Demo Coding vs Intergenomics (3) Demo Human vs Worm (4) Human Enhancers Cohn (5) Human Enhancers Ensembl (6) Human Ensembl Regulator and (7) Human OCR Ensembl. Each task involves classification between 2-3 class labels based on a genomic sequence provided. In each of these classification tasks, we construct data-constrained fine-tuning settings. We fine-tune a newly instantiated classifier head on a small subset of 2,048 training sequence examples and evaluate on 1,024 validation examples. We report the classification accuracy on the test set after task-specific fine-tuning as the primary evaluation metric.

**Genomic pre-training.** Before evaluating on the genomics tasks, we train each model on the Phase-1 OpenGenome2 corpus (Brixi et al., 2025) for 10,000 optimizer steps with the next token prediction objective (Brixi et al., 2025). We use a context length of 1,024, an effective batch size of 2,048 sequences (approximately 21B input tokens), and AdamW with learning rate $10^{-5}$, weight decay $0.1$, 10% linear warmup, cosine decay, and bfloat16 precision. Following Evo2 (Brixi et al., 2025), phylogenetic tags and sequence-stitch markers are retained as context but excluded from the loss, while lowercase repeat bases are mapped to their uppercase targets and down-weighted by $0.1$.

## 5 Results

In this section, we present downstream transfer results across different synthetic sources and pre-pre-training objectives.

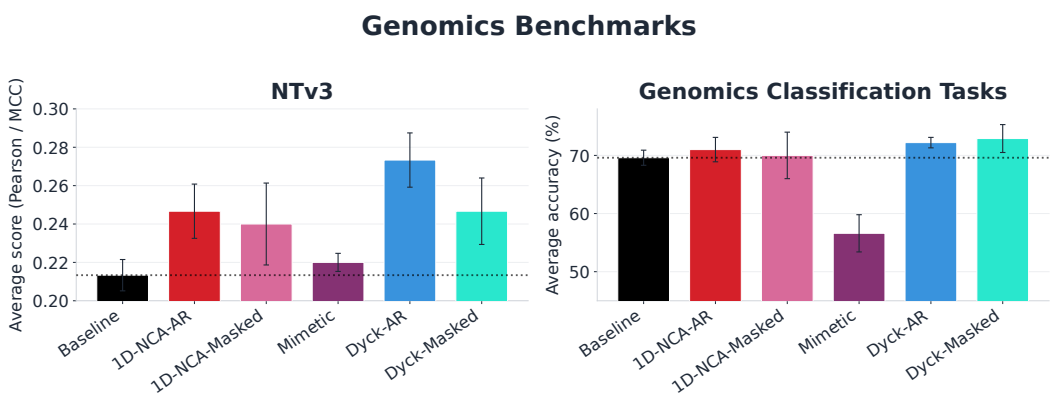



Figure 4: **Performance of synthetic pre-pretraining on genomics benchmarks.** We benchmark baseline genomics models against synthetic pre-pre-training variants across 10 downstream genomics tasks, evaluating both raw metric averages and the task-specific rank distributions. On the average evaluation metric, we find that Dyck models tend to perform the best for each benchmark.

Table 1: Detailed classification accuracy on genomic classification benchmarks by task and method. Values are mean ± sample standard deviation. Best results per task are shown in **bold**, and runner-up results are underlined.

| Task | Baseline | 1D-NCA-AR | 1D-NCA Masked | Mimetic | Dyck-AR | Dyck Masked |
|---|---|---|---|---|---|---|
| Human Non-Tata Promoters | 77.6 ± 1.3 | 77.9 ± 1.2 | 76.1 ± 2.4 | 69.5 ± 10.1 | 77.5 ± 1.8 | **79.0 ± 0.4** |
| Demo Coding vs Intergenomics | 72.8 ± 0.6 | 72.7 ± 0.9 | 71.3 ± 2.0 | 57.3 ± 9.2 | 74.9 ± 4.6 | **78.3 ± 5.2** |
| Demo Human vs Worm | 85.2 ± 6.5 | **89.3 ± 1.1** | 74.5 ± 14.5 | 53.4 ± 4.0 | 89.0 ± 2.1 | 89.2 ± 1.6 |
| Human Enhancers Cohn | 65.9 ± 0.4 | **66.5 ± 0.3** | 66.3 ± 0.5 | 55.7 ± 4.8 | 66.1 ± 0.3 | 65.3 ± 0.6 |
| Human Enhancers Ensembl | 71.1 ± 0.7 | 68.6 ± 1.2 | **72.1 ± 2.1** | 58.7 ± 7.5 | 69.3 ± 1.8 | 68.6 ± 2.0 |
| Human Ensembl Regulatory | 54.3 ± 1.6 | 65.4 ± 15.8 | **70.8 ± 13.4** | 49.6 ± 5.6 | 68.3 ± 7.8 | 70.4 ± 15.5 |
| Human OCR Ensembl | 60.0 ± 0.4 | 56.9 ± 0.9 | 58.6 ± 3.7 | 51.8 ± 2.0 | **60.3 ± 0.4** | 59.7 ± 1.9 |

### 5.1 NCA PRE-PRE-TRAINING WITH MASKED RECONSTRUCTION OBJECTIVE PERFORMS THE BEST ON VISION TASKS

We report the image classification accuracy of models across all evaluated families of synthetic data and baselines in Figure 3 across TinyImageNet, CIFAR-100 and STL-10. Across all benchmarks, we find that NCA pre-pre-training with the masked reconstruction objective outperforms all other methods and baselines. This indicates that synthetic priors learned from NCA pre-pre-training exhibit the strongest compatibility with vision among the families evaluated.

In both classes of synthetic data, Dyck and NCA, the ViT models pre-pre-trained with the masked modeling objective significantly outperforms the autoregressive objective. In fact, for Dyck formal languages, pre-pre-training on the autoregressive objective actually degrades model performance compared to the randomly initialized baseline. Taken together, these results suggest that as long as the synthetic source contains rich, predictable, structure, the masked reconstruction objective is best suited for learning transferable visual representations, echoing the success of masked autoencoders (He et al., 2021).

Prior work identifies attention as the primary carrier of transferable computational structure (Lee et al., 2026), making mimetic initialization a particularly informative baseline. Mimetic initialization reproduces characteristic weight structure in pretrained ViTs: a positive diagonal in $W_Q W_K^\top$ and a negative diagonal in $W_V W_{\text{proj}}$ (Trockman & Kolter, 2023). Yet, mimetic-initialized models trail NCA pre-pre-trained models by $\sim 3\%$ on average, with pronounced gaps in TinyImageNet and STL-10. Thus, this difference demonstrates that the benefits of synthetic pre-pre-training cannot be reproduced through improved attention initialization alone.

Table 2: Fine-tuning results across Human BigWig, Human Annotation, and Chicken BigWig benchmarks. For each model family, the complete column with the higher average performance across the three tasks is reported. Values are mean ± sample standard deviation across six seed triplets. Best results per task are shown in **bold**, and runner-up results are underlined; rankings are determined before rounding.

| Task | Baseline | 1D-NCA-AR | 1D-NCA Masked | Mimetic | Dyck-AR | Dyck Masked |
|---|---|---|---|---|---|---|
| Human BigWig Pearson | $0.29 \pm 0.01$ | $0.30 \pm 0.01$ | $0.30 \pm 0.01$ | $0.29 \pm 0.01$ | $\mathbf{0.32 \pm 0.01}$ | $0.29 \pm 0.01$ |
| Human Annotation MCC | $0.02 \pm 0.01$ | $0.09 \pm 0.04$ | $0.06 \pm 0.06$ | $0.04 \pm 0.01$ | $0.11 \pm 0.04$ | $\mathbf{0.11 \pm 0.05}$ |
| Chicken BigWig Pearson | $0.33 \pm 0.02$ | $0.35 \pm 0.01$ | $0.36 \pm 0.02$ | $0.33 \pm 0.00$ | $\mathbf{0.39 \pm 0.01}$ | $0.34 \pm 0.01$ |

### 5.2 Dyck and NCA priors transfer differently across genomic tasks

We report genomics performance across NTv3 and Genomics Classification in Figure 4, with more detailed task-level results in Tables 1 and 2. The benefits of synthetic pre-pre-training observed in vision also extend to genomics, where they remain evident after extensive natural-data training. Even after 21B tokens of OpenGenome2 pre-training and task-specific fine-tuning, improvements from synthetic pre-pre-training persist. We observe that synthetic pre-pre-trained models yield the highest reported mean on each of the ten downstream tasks. These results illustrate that synthetic pre-pre-training can complement relatively large-scale pre-training.

However, unlike in vision, genomics does not exhibit a single source–objective combination that dominates across all benchmark tasks. Transfer depends jointly on the synthetic substrate, pre-pre-training objective, and downstream task. Dyck in general provides the strongest average genomic transfer, while NCA retains task-specific advantages. Dyck-AR improves the mean NTv3 score over the randomly initialized baseline by approximately 6 points, and Dyck-Masked achieves the highest classification average of 72.9%, a gain of approximately 3.4% from the randomly initialized baseline. Yet, NCA variants achieve the highest mean accuracy on four of the seven classification tasks, indicating that Dyck pre-pre-training is not universally optimal.

The effect of the pre-pre-training evidently depends on the objective. Within Dyck, AR outperforms the masked objective on two of the three NTv3 tasks but only three of the seven classification tasks. Within NCA, AR is at least tied with the masked objective on nearly all NTv3 tasks and outperforms it on four of seven classification tasks. Genomics therefore does not exhibit a general preference for either autoregressive or masked modeling.

As in vision, we also find that mimetic initializations are unable to recover the performance of synthetic pre-pre-training, illustrating that performance gains are not easily recovered with better initializations of the attention heads.

## 6 Conclusion and Limitations

We demonstrate that structured, semantics-free pre-pre-training generalizes cross-domain to both vision and genomics. To our knowledge, this is the first study to systematically examine the the generality of the synthetic pre-pre-training framework across multiple domains, and we hope that this work serves as a promising initial map of this design space. The computational demands of these experiments constrain the scope of our study to two synthetic families, two objectives, and two domains. Broader studies across architectures, scales, and domains, and further investigation into the exact mechanisms of transfer from the synthetic to natural domains would be instrumental in building improved synthetic pre-pre-training recipes.

## AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing writing, developing figure aesthetics, and implementing and debugging some experimental code.

We have not used generative AI tools for generating synthetic datasets or generating literature review. Mathematical proofs are not applicable to this work. Additionally, we have not used generative AI tools to run experiments, write full portions of the text, or develop the core scientific ideas and methods of this work.

We have reviewed all AI-assisted work. Authors all checked LLM-generated figures and code for accuracy and correctness. All text was primarily handwritten, with AI usage only for rewording and styling. All literature review and references were manually done without generative AI assistance. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

Regarding the reproducibility of our work, we describe all the methods, training, and data details alongside hyperparameters used in our training. We plan to release our code for reproducing not only our methods but also our baseline methods that we compare against in this work.

## REFERENCES


Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations, 2021. URL https://arxiv.org/abs/2106.07682.

Manel Baradad, Richard Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. Procedural image programs for representation learning. *Advances in Neural Information Processing Systems*, 35:6450–6462, 2022.

Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

Sam Boshar, Benjamin Evans, Ziqi Tang, Armand Picard, Yanis Adel, Franziska K. Lorbeer, Chandana Rajesh, Tristan Karch, Shawn Sidbon, David Emms, Javier Mendoza-Revilla, Fatimah Al-Ani, Evan Seitz, Yair Schiff, Yohan Bornachot, Ariana Hernandez, Marie Lopez, Alexandre Laterre, Karim Beguir, Peter Koo, Volodymyr Kuleshov, Alexander Stark, Bernardo P. de Almeida, and Thomas Pierrot. A foundational model for joint sequence-function multi-species modeling at scale for long-range genomic prediction. *bioRxiv*, 2025. doi: 10.64898/2025.12.22.695963. URL https://www.biorxiv.org/content/early/2025/12/25/2025.12.22.695963.

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Mohsen Naghipourfar, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Gwanggyu Sun, Ali Taghibakshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Etowah Adams, Stephen A. Baccus, Steven Dillmann, Stefano Ermon, Daniel Guo, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R.K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, Jonathan C. Schmok, John St. John, Jeremy Sullivan, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modeling and design across all domains of life with evo 2. *bioRxiv*, 2025. doi: 10.1101/2025.02.18.638918. URL https://www.biorxiv.org/content/early/2025/02/21/2025.02.18.638918.

N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages*. In P. Braffort and D. Hirschberg (eds.), *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pp. 118–161. Elsevier. doi: https://doi.org/10.1016/S0049-237X(08)72023-8. URL https://www.sciencedirect.com/science/article/pii/S0049237X08720238.

Laure Ciernik, Lorenz Linhardt, Marco Morik, Jonas Dippel, Simon Kornblith, and Lukas Muttenthaler. Objective drives the consistency of representational similarity across datasets. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10920–10948. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/ciernik25a.html.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/coates11a.html.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. URL https://arxiv.org/abs/1909.13719.

Bernardo P. de Almeida, Hugo Dalla-Torre, Guillaume Richard, Christopher Blum, Lorenz Hexemer, Maxence Gélard, Javier Mendoza-Revilla, Ziqi Tang, Frederikke I. Marin, David M. Emms, Priyanka Pandey, Stefan Laurent, Marie Lopez, Alexandre Laterre, Maren Lang, Uğur Şahin, Karim Beguir, and Thomas Pierrot. Annotating the genome at single-nucleotide resolution with DNA foundation models. 22(11):2301–2315, 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02881-2. URL https://doi.org/10.1038/s41592-025-02881-2.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

Fabian Gröger, Shuo Wen, and Maria Brbić. Revisiting the platonic representation hypothesis: An aristotelian view, 2026. URL https://arxiv.org/abs/2602.14486.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. URL https://arxiv.org/abs/2111.06377.

Michael Y Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases. *arXiv preprint arXiv:2502.19249*, 2025a.

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

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis, 2024. URL https://arxiv.org/abs/2405.07987.

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

Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

A. Sophia Koepke, Daniil Zverev, Shiry Ginosar, and Alexei A. Efros. Back into plato's cave: Examining cross-modal representational convergence at scale, 2026. URL https://arxiv.org/abs/2604.18572.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. 2015. URL https://api.semanticscholar.org/CorpusID:16664790.

Dan Lee, Seungwook Han, Akarsh Kumar, and Pulkit Agrawal. Training language models via neural cellular automata, 2026. URL https://arxiv.org/abs/2603.10055.

Jiaang Li, Yova Kementchedjhieva, Constanza Fierro, and Anders Søgaard. Do vision and language models share concepts? a vector space alignment study. *Transactions of the Association for Computational Linguistics*, 12: 1232–1249, 2024. doi: 10.1162/tacl_a_00698. URL https://aclanthology.org/2024.tacl-1.68/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Mayug Maniparambil, Raiymbek Akshulakov, Yasser Abdelaziz Dahou Djilali, Sanath Narayan, Mohamed El Amine Seddik, Karttikeya Mangalam, and Noel E. O'Connor. Do vision and language encoders represent the world similarly?, 2024. URL https://arxiv.org/abs/2401.05224.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation, 2018. URL https://arxiv.org/abs/1806.05759.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. https://distill.pub/2020/growing-ca.

Isabel Papadimitriou and Dan Jurafsky. Learning Music Helps You Read: Using transfer to study linguistic structure in language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6829–6839, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.554. URL https://aclanthology.org/2020.emnlp-main.554/.

Geoffrey Roeder, Luke Metz, and Diederik P. Kingma. On linear identifiability of learned representations, 2020. URL https://arxiv.org/abs/2007.00810.

Zachary Shinnick, Liangze Jiang, Hemanth Saratchandran, Damien Teney, and Anton van den Hengel. Can you learn to see without images? procedural warm-up for vision transformers, 2025. URL https://arxiv.org/abs/2511.13945.

Megan Tjandrasuwita, Chanakya Ekbote, Liu Ziyin, and Paul Pu Liang. Understanding the emergence of multimodal representation alignment, 2025. URL https://arxiv.org/abs/2502.16282.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers  distillation through attention, 2021. URL https://arxiv.org/abs/2012.12877.

Asher Trockman and J. Zico Kolter. Mimetic initialization of self-attention layers, 2023. URL https://arxiv.org/abs/2305.09828.

Stephen Wolfram. 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.

Yuhuai Wu, Markus Rabe, Wenda Li, Jimmy Ba, Roger Grosse, and Christian Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning, 2022. URL https://arxiv.org/abs/2101.06223.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019. URL https://arxiv.org/abs/1905.04899.

Table 3: NCA pre-pre-training configuration. Both settings use the same generator, tokenization, and optimizer; they differ in grid geometry, alphabet size, and which objective matches the downstream attention pattern.

| | **2D (vision)** | **1D (genomics)** |
|---|---|---|
| States $n$ / grid $g$ | 2 / $32 \times 32$ torus | 20 / ring of 4096 |
| Neighborhood $k$ | 3 | swept: 3, 6, 7, 9 |
| Temperature $\tau$ / burn-in $t_0$ | $10^{-4}$ / 10 | $10^{-4}$ / 10 |
| Frames per sequence $T$ | 16 | 1 |
| Patch $p$ / patch vocabulary | 4 / $2^{16}$ | 4 / $20^4$ |
| Context length $L$ | 1024 | 1024 |
| gzip band $[r_{\text{lo}}, r_{\text{hi}}]$ | $[0, 1]$ | $[0.5, 1]$ |
| Rules per epoch (train / held-out) | 16,000 / 2,000 | 4,992 / 2,000 |
| Trunk | 12L, 3H, 192d (5.5M) | 24L, 16H, 1024d (0.40B) |
| Objective | next-token and masked ($m$=0.3) | next-token |
| Optimizer | AdamW, lr $10^{-4}$, no weight decay, cosine, 10-epoch warmup | |
| Batch | 32 sequences (≈33k tokens) per step | |
| Duration | 300 epochs | 100 epochs |

Table 4: Dyck pre-pre-training configurations for vision and genomics transfer. The checkpoint row reports the number of optimizer updates used before transfer; the schedule horizon specifies the full learning-rate schedule. Masking probabilities are conditional on closing brackets.

| | **Vision** | | **Genomics** | |
|---|---|---|---|---|
| | **AR** | **Masked** | **AR** | **Masked** |
| Bracket types | 128 | 64 | 128 | 128 |
| Bracket vocabulary | 256 | 128 | 256 | 256 |
| Input vocabulary, including special tokens | 256 | 130 | 256 | 257 |
| Context length $L$ | 1024 | 196 | 1024 | 1024 |
| Opening probability | 0.5 | 0.6 | 0.5 | 0.5 |
| Training corpus size | 160,000 | generated online | 1,600,000 | 1,600,000 |
| Held-out sequences | 10,000 | — | 10,000 | 10,000 |
| Trunk | 12L, 3H, 192d | | 24L, 16H, 1024d | |
| Attention | causal | bidirectional | causal | bidirectional |
| Mask probability over closing brackets | — | 0.5 | — | 0.5 |
| Optimizer | AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ | | | |
| Peak learning rate | $10^{-4}$ | $2 \times 10^{-3}$ | $10^{-4}$ | $10^{-4}$ |
| Weight decay | 0 | 0.05 | 0.1 | 0.1 |
| Learning-rate schedule | linear warmup followed by cosine decay | | | |
| Warmup updates | 5,000 | 1,000 | 500 | 500 |
| Schedule horizon (updates) | 50,000 | 15,000 | 5,000 | 5,000 |
| Global batch size (sequences) | 32 | 256 | 32 | 32 |
| Frozen embeddings | none | token and position | none | none |
| Checkpoint used for transfer (updates) | 5,000 | 15,000 | 5,000 | 5,000 |

# A Pre-pre-training Hyperparameters

We detail the NCA and Dyck pre-pre-training configurations in Tables 3 and 4. During pre-pre-training, we sweep embedding weight re-initializations, learning rates ranging from $10^{-3}$ to $10^{-5}$, and various weight decays ranging from 0 to 0.1. We evaluate hyperparameters based on performance on the validation pre-pre-training data and transfer to the target domain with early stopping. We do not conduct an extensive sweep of specific substrate-specific configurations for Dyck (e.g. number of brackets, depth) and NCA pre-pre-training (e.g. color, grid size, complexity), and leave this to future work.

# B GENOMICS EVALUATION IMPLEMENTATION DETAILS

For NTv3Bench and Genomics Classifications tasks, we report using two independent pre-training seeds crossed with three downstream fine-tuning seeds for each model configuration. We detail the precise implementation and training compute used for these downstream tasks.

## B.1 NTV3BENCH

We fully fine-tune the genomic backbone and a newly initialized prediction head. Predictions are evaluated over the central 12,288 positions, with the remaining sequence serving as context. Training uses AdamW with weight decay $0.01$, an effective batch size of 32, and bfloat16 precision. We warm up the learning rate linearly from $10^{-5}$ to $5 \times 10^{-5}$ during the first 3% of training and then decays to $2.5 \times 10^{-5}$. Human tasks are trained for 19,932 optimizer steps (20.9B input tokens), while chicken functional-track prediction uses 9,033 steps (9.47B tokens). Validation is performed every 1,000 steps on 1,000 examples.

For functional-track prediction, we optimize the Poisson–multinomial loss, with shape-loss coefficient 5. The human and chicken models predict 34 and 14 tracks, respectively, and checkpoints are selected by mean validation Pearson correlation. For human genome annotation, we use focal loss with $\gamma = 2$ and select checkpoints by mean validation MCC. Annotation validation and test sets contain 1,000 and 10,000 non-overlapping windows, respectively; functional-track evaluation uses the complete benchmark test split.

We use 8 H200 GPUs for training. The human functional-track prediction and human genome annotation each take approximately 140 GPU hours per run. The chicken functional prediction takes 64 GPU hours per run.

## B.2 GENOMICS CLASSIFICATION TASKS

Across each task family, we fully fine-tune the genomics backbone and a newly initialized classifier head. Training uses AdamW with a weight decay of 0.01, a batch size of 8, and bfloat 16 precision. We use a constant learning rate of $3 \times 10^{-5}$. Each task is trained for a single epoch on the example training set of $2,048$ resulting in $256$ optimizer steps. For each task, the evaluation is done on the $1,024$ test set per task.

We run each of these evaluations on a single A100 or H200 GPU with training and evaluations taking approximately half an hour per run.