This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Can You Learn to See Without Images? Procedural Warm-Up for Vision Transformers

Zachary Shinnick*[1] Liangze Jiang[2,3] Hemanth Saratchandran[1]
Damien Teney[3] Anton van den Hengel[1]

[1]Australian Institute for Machine Learning (AIML), University of Adelaide, Australia
[2]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[3]Idiap Research Institute, Switzerland

Abstract

Transformers are remarkably versatile, suggesting the existence of generic inductive biases beneficial across modalities. In this work, we explore a new way to instil such biases in vision transformers (ViTs) through pretraining on procedurally generated data devoid of visual or semantic content. We generate this data with simple algorithms such as formal grammars, so the results bear no relationship to either natural or synthetic images. We use this procedurally generated data to pretrain ViTs in a warm-up phase that bypasses their visual patch embedding mechanisms, thus encouraging the models to internalise abstract computational priors. When followed by standard image-based training, this warm-up significantly improves data efficiency, convergence speed, and downstream performance. On ImageNet-1K, for example, allocating just 1% of the training budget to procedural data improves final accuracy by over 1.7%. In terms of its effect on performance, 1% procedurally generated data is thus equivalent to 28% of the ImageNet-1K data. These findings suggest a promising path toward new data-efficient and domain-agnostic pretraining strategies.

Project page & code: *https://zlshinnick.github.io/procedural-pretraining-page/*

1. Introduction

Machine learning (ML) is moving towards increasingly generic approaches. While early efforts in ML were often domain-specific, recent work shows evidence of learning methods [14, 48], computational mechanisms [16, 22, 44], and representations [21] that are applicable across tasks and modalities. One prominent example is the transformer architecture [52], which implements generic inductive biases applicable to language as well as vision and other domains. Besides architectural choices, acquiring domain-agnostic mechanisms *from data* could offer greater flexibility. Yet this possibility remains underexplored. Existing examples include training large language models (LLMs) on computer code as a means to improve reasoning [37], and the observation that models trained on text acquire capabilities for handling visual data [16, 43]. These examples suggest the existence of generic mechanisms independent from any domain or modality.

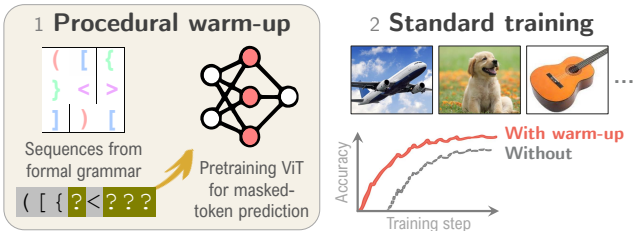


Figure 1. We propose a new pretraining phase for ViTs using synthetic data that is symbolic and non-visual. **(Left)** We generate procedural data consisting of sequences of abstract tokens using formal grammars (e.g. sequences of balanced parentheses). We bypass the ViT's visual patch embedding and directly train the model for masked-token prediction. **(Right)** Followed by standard image-based training (e.g. on ImageNet), our lightweight warm-up consistently improves convergence and downstream performance.

Improving language models with abstract data. Recent work shows that language models can benefit from abstract data that *imitates* the structure of natural language, such as with procedural data [1] sampled from formal grammars [19]. This data is devoid of any semantic information, yet it can improve the convergence and performance of large language models (LLMs). Other works show similar benefits with procedural data generated with other simple algorithms [5, 22, 55] and cellular automata [57] that can teach the model generic computational mechanisms.

*Correspondence to: zachary.shinnick@adelaide.edu.au.

[1]*Procedural data* refers to data generated with explicit algorithms, whereas *synthetic data* typically refers to data from trained models, e.g. text from LLMs [30, 38] or images from GANs and diffusion models [1, 39].

Can vision models learn without images? We hypothesize that vision models could also benefit from procedural data, as a means to acquire generic mechanisms relevant to vision tasks. However, it is unclear how vision models can 'see' without images. Indeed, there is a rich literature on the training of vision models on synthetic and/or abstract images [18, 39, 41, 45, 50]. These works typically rely on handcrafted algorithms that generate abstract images imitating properties of natural data [42]. These images have mostly been evaluated as a replacement for standard datasets to address privacy and fairness concerns. In this paper, we go a step further and completely forgo the constraints of visual data. Instead, we train vision models on procedural data with no 2D structure or explicit correspondence with image properties. The hope is that this data might capture even more generally useful structure as a result. The problem of reasoning over images is primarily a reasoning problem, not an image problem.

Proposed method and results. We propose a training pipeline for vision transformers (ViTs) that leverages procedural data in a brief warm-up prior to standard training on natural images (see Figure 1). Our procedural data is generated with simple formal grammars, contains no semantic information, and does not attempt to imitate visual structures. This data is cheap to generate and process. To feed it to ViTs, we bypass the visual patch embeddings, and map instead the abstract symbols to random frozen embeddings, which are later discarded (see Section 3). We conduct an extensive evaluation of the value of this abstract data for image classification tasks. We generally find substantial improvements in downstream performance. Allocating as little as 1% of the total training budget to procedural data yields a +1.7% improvement in accuracy on IMAGENET-1K. This means that it can replace up to 28% of the image data for equivalent performance. The improvements also persist when the model is fine-tuned on various domain-specific datasets. This suggests that the information learned from procedural data is distinct and complementary to standard visual learning, rather than an analogue or substitute.

Detailed analysis. With careful ablations, we verify that the benefits arise from structured dependencies within the procedural data. We also find that procedural warm-up acts both on attention and MLP layers, contrasting with existing structured initializations of transformers, which act only on the attention weights [13, 51, 60]. Procedural warm-up also acts mainly on the *latter* layers, which contrasts with the established knowledge that *early* layers are most important for standard visual pretraining [27]. This supports the conclusion that procedural warm-up provides a qualitatively different type of training signal than standard data.

Our contributions are summarized as follows.

- **Procedural warm-up for vision transformers.** We propose a lightweight pretraining stage for ViTs using symbolic, non-visual data generated with formal grammars, as a means to acquire generic mechanisms that support subsequent visual learning. This expands recent work on procedural pretraining [19, 22] from LLMs to ViTs.
- **Empirical evaluation.** We implement our approach with standard ViTs and evaluate it on various benchmarks. We show that a brief procedural warm-up consistently accelerates convergence and improves the accuracy of the models. The gains persist when combined with large-scale pretraining on ImageNet, suggesting that procedural warm-up and visual pretraining provide complementary training signals.
- **Analysis of the source of improvements.** We show that the benefits arise from the precise structure of the procedural data, and manifest in both attention and MLP layers, particularly in later layers. This surprising observation contrasts with standard pretraining which acts primarily on *early* layers. This suggests that procedural warm-up is qualitatively different from standard visual pretraining or existing structured initialization methods.

2. Related Work

Training vision models with abstract images. The properties of natural images have been extensively studied [42]. Many procedural algorithms exist to generate images that imitate structural and statistical properties of natural data [3]. Abstract images used to train vision models include fractals [33, 34], contours [24], wave patterns [47], and structured noise [2]. The motivation is usually to replace natural images for privacy and fairness reasons [23], or to allow learning from smaller datasets [35]. Our work differs because we use *non-visual* data, and our goal is to *complement* natural images by providing a different training signal.

Vision capabilities from non-visual data. Recent work shows that some text-only models are capable of processing visual information [11, 59], sometimes after minimal adaptation [28, 49]. Han et al. show that these effects arise from the general knowledge present in text data (e.g. descriptions of visual attributes) but also from the transfer of modality-agnostic reasoning mechanisms. This motivates our attempt to improve vision capabilities from non-visual data.

Initialization of vision models. The initial weights of neural networks are critical for the convergence of gradient-based training [20]. Besides simple adjustments to the distribution of random weights, structured initializations exist to improve the training speed or generalization of transformers [20, 51, 58]. The *Mimetic* initialization creates diagonal patterns in attention matrices that imitate trained models. Other methods initialize the attention of ViTs to imitate the behaviour of CNNs [13, 60]. These approaches affect only the attention layers, while our method acts on the entire network. They all aim to instantiate structures presumed to be useful for subsequent vision tasks.

Table 1. **Formal languages used to generate our procedural data.** We select standard simple grammars representative of different levels of structural complexity in the Chomsky hierarchy, from regular (WW) to context-free (k-DYCK) and context-sensitive (k-DYCK SHUFFLE).

Language	Type	Example sequence	Structural property
WW	Regular	a b c a b c	Global one-to-one correspondence
k-DYCK	Context-free	([ ]) < >	Hierarchical, stack-based dependencies
k-DYCK SHUFFLE	Context-sensitive	([ ) <] >	Crossing and interleaved dependencies

Procedural data in other domains. The NLP literature (natural language processing) is rich in attempts to train models with abstract synthetic data e.g. for machine translation [17], mathematical reasoning [54], text encoding [40], and general-purpose language modeling. Formal grammars are a common approach to sample data devoid of semantic meaning that imitates properties of natural language [6, 15, 19, 32, 36, 40]. Recent work also uses data generated with other simple algorithms including cellular automata [29, 54, 55, 57]. Most similar to our work, [22] and [44] propose a procedural warm-up stage for LLMs, and find that it provides training information complementary to standard datasets, which yields improved performance with natural language, computer code, and informal mathematics. Finally, procedural data has also been used to train models for visual navigation [53], and reinforcement learning [3].

3. Proposed Method

This section presents our method for procedural warm-up of ViTs. First, we generate procedural data with formal languages (Section 3.1). This data is symbolic, non-visual, and easily predictable given the knowledge of the generating grammar. Second, we use this data in a *lightweight* pretraining stage where a ViT is trained for standard masked-token prediction (Section 3.2). This drives the model to discover the data-generating algorithm and thereby learn generic computational mechanisms. We will show in Section 4 that the resulting model is a much better starting point for standard image-based training than a typical random initialization, improving both its training speed and final performance.

3.1. Generating Procedural Data

Our goal is to cheaply generate data that contains predictable structures with different levels of complexity. Formal languages offer a simple specification of a flexible set of data-generating algorithms. Moreover, data generated with formal languages has been used with success in LLMs [19, 22] as a complement to natural language.

Selected formal grammars. For our study, we select a set of simple grammars summarized in Table 1. They span multiple levels of complexity in the *Chomsky hierarchy* [7]. The levels of this hierarchy correspond to the computational machinery necessary to recognise these languages: *regular* languages can be processed by finite-state automata, *context-free* languages require a stack to capture hierarchical structure, and *context-sensitive* languages extend this further to represent cross-serial and multi-level dependencies. Consequently, data generated with our chosen grammars contain a variety of structures, from simple regular repetition to nested and cross-serial dependencies:

- **WW.** A simple *regular* language consisting of a string concatenated with its exact copy.
- **k-DYCK.** A *context-free* language that generates balanced parentheses with a hierarchical structure. The parameter k controls the number of unique bracket pairs, which determines the variety and complexity of the nesting.
- **k-DYCK SHUFFLE.** Relaxes the k-DYCK grammar by removing the constraint that brackets must be well-nested, while preserving the requirement that every opening bracket is properly closed. This yields a *context-sensitive* language with crossing dependencies, representing multiple intertwined hierarchies.

Generating token sequences for ViTs. For any chosen grammar, we generate sequences using a stochastic generator matched to its structure: WW sequences are formed by sampling a random substring and concatenating it with an exact copy; k-DYCK strings are produced with a stack-based process that randomly opens and closes brackets while enforcing proper nesting; and k-DYCK SHUFFLE sequences are generated by stochastically opening new brackets or closing currently open ones. We use a vocabulary of 128 tokens (unless stated otherwise), following [19], which means setting $k = 64$ for the Dyck languages. ViTs are designed to take in a fixed number of visual tokens, thus, each procedural sequence is generated with a fixed size of $N{=}H \times W$. Importantly, this symbolic data can be generated at a negligible computational cost, enabling a lightweight warm-up stage with no natural images.

3.2. Pretraining ViTs with Procedural Data

Feeding abstract tokens to ViTs. Our study uses standard ViTs [10], which start with a linear projection known as the patch embedding layer. This normally projects the image contents to a sequence of feature vectors. Since our procedural data is made of tokens from a discrete vocabulary, we use instead a discrete embedding layer as found in language models (i.e. a lookup table). We keep this embedding fixed with random vectors, such that the symbols from the vocabulary are assigned to approximately orthogonal vectors. This

prevents the model from solving the pretraining objective through the embeddings, forcing it instead to adapt its attention and MLP layers. We similarly maintain the positional encoding frozen during procedural pretraining.

Pretraining objective. The procedural warm-up uses a standard masked-token objective with a token prediction head. For each sequence, we mask 50% of structurally informed tokens, such as closing brackets in DYCK and DYCK SHUFFLE, or the repeated sequence in WW, and require the model to predict the original token. As mentioned above, we keep the embedding layer frozen during the procedural pretraining and only update the attention and MLP layers.

Standard image-based training. After the procedural warm-up, we proceed with standard pretraining and/or finetuning on natural images. We discard the token embeddings and prediction head that were necessary for the procedural data. We then update the entire architecture using standard learning rates and hyperparameters. The procedurally-pretrained model can thus be compared to other forms of weight initialization.

Note on ambiguous grammars. For the k-DYCK and WW languages, we use a loss mask such that each example has a unique valid completion (masking opening parentheses and the first half of sequences, respectively). For k-DYCK SHUFFLE, each sequence admits multiple valid completions. For example, '`( [ Mask < Mask >`' could be completed with '`( [ ] < ) >`' or '`( [ ) < ] >`'. We train the model with "teacher-forcing" supervision on one valid continuation for each training example. When trained on this data, the model thus cannot reach perfect accuracy, yet it drives the model to learn to track stack depth and bracket types to assign non-zero probability to valid continuations. Overall, learning valid completions for our formal languages requires learning and implementing generic mechanisms for hierarchical composition, long-range dependencies, and stack manipulation.

Summary. Our procedural warm-up (i) uses procedurally generated symbolic data from formal languages, (ii) freezes the embedding layers, ensuring that the model encodes new knowledge in its attention and MLP layers, and (iii) uses the pretrained weights as initialization for standard image-based training. The setup allows us to acquire knowledge from abstract symbolic data and leverage it for vision tasks. The next section empirically evaluates the value of this abstract knowledge on a variety of use cases and datasets.

4. Experiments

Our goal here is to evaluate the value of procedural pretraining with ViTs on standard image classification tasks. The experiments are designed to address the following questions.

1. Do ViTs acquire information from our abstract non-visual data that is useful for vision tasks?
2. Are the effects of procedural warm-up distinct and complementary to standard large-scale visual pretraining?
3. What properties of the procedural data are responsible for the observed benefits?
4. How is the learned information represented within the model's layers?

Experimental setup. We use a standard ViT-B/16 architecture for experiments with the IMAGENET-1K dataset and a ViT-T/16 for the smaller datasets [10]. We expose the models to our procedural warm-up phase as outlined in Section 3 before standard training on natural images. We use the same hyperparameters throughout the procedural warm-up and the standard training phases, with common choices of learning rate, AdamW optimizer, and cosine learning-rate decay (see details in the appendix).

Datasets. We use standard datasets that vary in size and complexity to evaluate performance on image classification: IMAGENET-1K [9], TINY-IMAGENET [26], FOOD-101 [4], CIFAR-10 and CIFAR-100 [25], and STL-10 [8]. With IMAGENET-1K, we also evaluate the complementarity of procedural warm-up and conventional large-scale visual pretraining.

Baselines. Our method is meant to provide a generic starting point for training ViTs. Therefore, we compare it to alternative initialization strategies also designed to improve subsequent image-based training.

- **Default initialization [10]:** standard sampling of random weights from a truncated-normal distribution.
- **Mimetic initialization [51]:** a structured initialization scheme for attention layers that imitates a diagonal-matrix structure observed in pretrained models.
- **FractalDB warm-up (sample-matched) [23]:** a form of procedural *visual* data also designed to train vision models. It was originally intended as a sole source of pretraining data for visual representation learning, rather than a complement like ours. To match our setting, first, we adjust and match the number of training examples, and second, we reset the patch-embedding and classifier layers after pretraining as described in Section 3.2. This allows a fair comparison of procedural-visual data (FractalDB) vs. procedural-symbolic data (ours).

These methods are followed with standard training with identical optimization settings, such that subsequent differences in performance arise solely from the initialization or warm-up applied to the ViT.

4.1. Improving Vision without Images

Background. We first evaluate whether a ViT can acquire information that is useful for vision tasks from our procedural non-visual data. Recent work already shows that this kind of data can help with language, code, and mathematics [22], and we now test whether these effects extend to vision.

Table 2. **Accuracy (%) on image-classification benchmarks.** Models subject to procedural warm-up consistently converge to a higher accuracy than other generic initialisation/pretraining schemes. Green subscripts denote absolute improvements over the default initialization.

Method	IMAGENET-1K	TINY-IMAGENET	FOOD-101	CIFAR-10	CIFAR-100	STL-10
Default random initialization [10]	77.49	55.42	74.52	91.29	68.52	60.52
Mimetic initialization [51]	78.68 +1.19	57.20 +1.78	79.21 +4.69	**92.89 +1.60**	70.72 +2.20	65.37 +4.85
FractalDB warm-up [23]	78.06 +0.57	55.17 -0.25	74.25 -0.27	88.98 -2.31	64.61 -3.91	58.62 -1.90
Procedural warm-up (ours)	**79.21 +1.72**	**58.20 +2.78**	**79.47 +4.95**	92.81 +1.52	**71.98 +3.46**	**66.48 +5.96**

Table 3. **Accuracy (%) of ViT-B models pretrained on IMAGENET-1K and fine-tuned on various datasets.** The procedural warm-up consistently improves the transfer performance compared to the default, Mimetic and FractalDB baselines. Green subscripts denote absolute improvements over the default ImageNet initialization. The fact that the improvements persist throughout large-scale pretraining and fine-tuning stages suggests that the procedural data provides a training signal distinct and complementary to natural images.

Method	TINY-IMAGENET	FOOD-101	CIFAR-10	CIFAR-100	STL-10
Random init. [10] + IMAGENET-1K	86.59	89.64	98.59	87.54	98.55
Mimetic init. [51] + IMAGENET-1K	87.29 +0.70	90.74 +1.10	**98.68 +0.09**	88.78 +1.24	**98.81 +0.26**
FractalDB [23] + IMAGENET-1K	**88.42 +1.83**	90.13 +0.49	98.41 -0.18	88.35 +0.81	98.46 -0.09
Procedural warm-up (ours) + IMAGENET-1K	87.93 +1.34	**90.79 +1.15**	**98.68 +0.09**	**89.20 +1.66**	98.66 +0.11

Setup. We apply a procedural warm-up to a ViT with the k-DYCK language ($k=64$, following the setup of [19]). This drives the model to discover a structure of nested dependencies. The warm-up is run for a fixed budget equivalent to 1% of the total IMAGENET-1K examples used in Sections 4.1–4.2, after which we proceed with standard training on image datasets as described in Section 3. This procedural warm-up can be seen as an alternative to standard random or structured initializations [20, 51, 58] because its computational cost is minimal and meant to be amortised over many trained models. We further study the computational budget of the warm-up in Section 4.3.

Results. Table 2 shows that the procedural warm-up consistently improves downstream performance across all benchmarks. On average, procedural warm-up yields a +3.4% absolute improvement over the default initialization. Compared to the Mimetic initialization, the procedural warm-up produces larger improvements, suggesting that our data provides a richer inductive bias than the structured-attention patterns instilled through the Mimetic initialization. Our method also outperforms the sample-matched FractalDB, despite the latter containing *visual* data. This is a very interesting observation, since the FractalDB data was specifically designed to capture structures relevant to visual tasks, whereas our data is much more generic and has no obvious correspondence with visual properties.

Importantly, we observe similar improvements on IMAGENET-1K with a larger ViT-B model, with a +1.72% improvement in top-1 accuracy. This means that the procedural warm-up is useful even when the dataset and model size increase. This suggests that the procedural warm-up and the standard training on natural images provide training signals that do not completely overlap nor override one another.

Take-away. ViTs can acquire useful information for vision tasks without images. Procedural warm-up on a generic k-DYCK language creates inductive biases beneficial to a broad range of image classification datasets.

4.2. Complementarity with Standard Training

Background. Section 4.1 shows that procedural warm-up helps prepare ViTs for one round of training on various target datasets. We now evaluate whether it helps when the ViTs also undergo standard pretraining on IMAGENET-1K, and whether the benefits persist through subsequent fine-tuning on target datasets. Prior work with language models [19, 22] already shows that procedural data can introduce persistent, complementary structure that survives pretraining on real language data.

Setup. We first expose the ViT to a small amount of procedural k-DYCK data ($k=64$), using E_1 samples (approximately 3.8 M symbolic examples, corresponding to 1% of the pretraining budget). The ViT then undergoes standard pretraining on E_2 samples from IMAGENET-1K (approximately 385 M image examples).

We consider two settings for combining procedural warm-up and visual pretraining: **additive** and **substitutive**. In the *additive* setting, E_2 is held fixed and we introduce E_1 to test whether procedural warm-up improves the quality of the resulting pretrained model beyond what large-scale visual pretraining alone provides. The resulting models are then fine-tuned on various benchmarks to assess the impact of the warm-up on final downstream performance.

In the *substitutive* setting, we reduce E_2 while introducing E_1, and measure the reduction ΔE_2 for which training on $E_2 - \Delta E_2$ natural images with E_1 procedural examples

matches the performance of the E_2-only model. This quantifies how procedural data can substitute for real images without degrading performance. All inputs have the same size, so the number of training examples and tokens are proportional to one another.

Results. Procedural warm-up provides a clear and complementary gain on top of large-scale visual pretraining. As shown in Figure 2, models initialized with a procedural warm-up follow a distinct optimization trajectory relative to default initialization, and ultimately reach a higher accuracy.

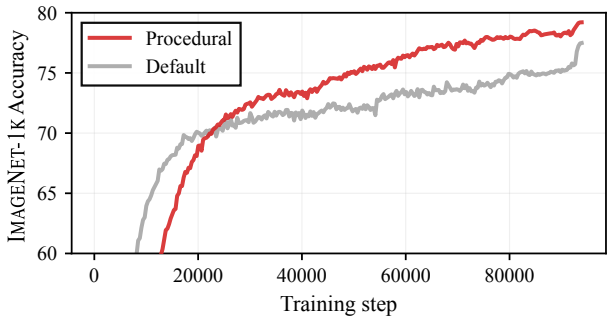


Figure 2. **Procedural warm-up leads to a distinct optimization trajectory.** With ImageNet pretraining held fixed, a model initialized with a brief procedural warm-up (red) shows a clearly distinct training curve than a default initialization (gray). This suggests that the procedural warm-up provides a qualitatively different training signal rather than merely a head-start on standard pretraining.

Additive setting. With ImageNet pretraining held fixed, Table 3 shows that our procedural warm-up improves performance across benchmarks and outperforms Mimetic/FractalDB. This indicates that the effects *persist after fine-tuning* and *complement standard visual pretraining*.

Substitutive setting. Figure 3 shows that replacing only *1% of the total pretraining budget* with procedural data yields models that match the performance of those trained solely on ImageNet-1k while using *28% fewer image examples*.

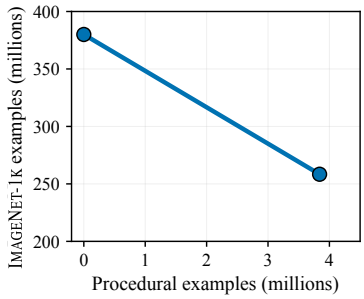


Figure 3. **Procedural warm-up reduces the ImageNet-1k data requirements.** Replacing 1% of the total pretraining budget with 3.8 M procedural samples allows the model to match the accuracy of a full ImageNet-1k pretraining while using 28% fewer natural-image samples (about 108 M fewer images).

Take-away. Procedural warm-up provides inductive biases that are *distinct* and *complementary* to standard large-scale visual pretraining, resulting in more efficient training and enhanced performance.

4.3. What are the Important Properties of the Procedural Data?

Background. The previous sections show that procedural warm-up can speed up the training and improve the performance of ViTs. We now seek to identify which properties of the procedural data are responsible for these effects.

Setup. We start with the k-Dyck ($k=64$) configuration and vary one property at a time. Specifically, we examine:

1. **Language type:** we test other data-generating grammars from Table 1, namely WW and k-Dyck Shuffle. These alternatives sit at different levels of the Chomsky hierarchy and are thus representative of clearly different structural complexity.
2. **Order preservation:** we retain the overall distribution of symbols in the generated set of procedural examples, but we shuffle the order of the tokens within every sequence to break their structural dependencies. This assesses whether the gains stem from precise structure or low-order statistics alone.
3. **Pretraining length:** we vary the number of gradient steps of the procedural warm-up.

Language type. Table 4 shows that the warm-up with both Dyck or Dyck Shuffle yields clear improvements over the default initialization, whereas WW offers no benefit. The strongest gains come from the context-free Dyck grammar, suggesting that a stack-based hierarchy imparts the most transferable inductive bias for vision. The regular WW language, which lacks nested structure, fails to help, indicating that simple repetition does not lead to learning useful mechanisms. The context-sensitive Dyck Shuffle improves performance but lags behind Dyck, implying that excessive structural entanglement weakens transferability.

Order preservation. Shuffling the token order within Dyck sequences eliminates the benefit of procedural warm-up (Table 5), reducing performance below the default initialization. This confirms that the gains arise from *structural order* rather than simply the token distribution. This shows that the hierarchical dependencies drive the inductive bias, rather than symbol frequency or co-occurrences.

Procedural pretraining length. As shown in Figure 4, downstream performance peaks at an intermediate number of procedural warm-up steps, with both shorter and longer runs yielding lower downstream performance. This pattern aligns with prior findings with language models [46], where excessive pretraining can hinder fine-tuning adaptability.

Table 4. **Effect of different formal languages.** Both DYCK and DYCK SHUFFLE improve over the baseline, while WW provides no benefit. Context-free structure yields the strongest gains, suggesting that hierarchical structured data is key to the observed benefits.

Method	Language type	CIFAR-100 (%)
Random initialization	—	68.52
WW	Regular	66.44
k-DYCK	Context-free	**71.98**
k-DYCK SHUFFLE	Context-sensitive	70.11

Table 5. **Effect of removing hierarchical structure in procedural data.** We shuffle tokens within DYCK sequences to remove hierarchical dependencies while preserving the token distribution. This degrades the performance, falling below a random initialization.

Method	CIFAR-100 (%)
Random initialization	68.52
k-DYCK	**71.98**
k-DYCK (shuffled sequences)	67.22

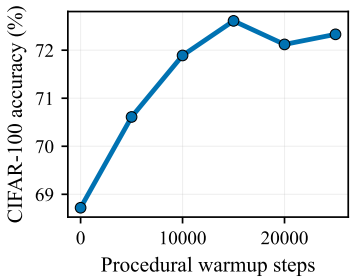


Figure 4. **Downstream accuracy as a function of the number of warm-up steps.** The accuracy peaks at an intermediate value. With excessive pretraining, the model likely over-specializes to the procedural data, which hinders adaptation to visual tasks.

Take-away. Procedural warm-up benefits ViTs when the data contains hierarchical structure, when input order is exact, and when training is neither too short nor excessive.

4.4. Which Layers Contain Useful Information from Procedural Warmup?

Background. The previous sections identified the effects of procedural warm-up and the responsible properties of the data. We now shift focus to the model itself: where and how are the learned inductive biases expressed within the ViT architecture? Methods like the Mimetic initialization [51] act solely on the *attention weights*, while our procedural warm-up can also affect the MLP layers.

Setup. We conduct three analyses to understand how the information learned during the procedural warm-up manifests within the network.

- **Weight structure.** We test whether the benefit depends on precise structure or merely the magnitude of the weights. We shuffle the weight values within a layer, which preserves the distribution of magnitudes while removing any precise structure.
- **Component contributions.** We test whether the useful information is distributed or located within specific components. To do so, we keep the correct weights from the procedural warm-up only for the attention or MLP layers, and shuffle the weights within each layer of the other type.
- **Layerwise analysis.** We identify which layers contribute most by transferring weights from the procedural warm-up only for the early, middle, or late $1/3^{\text{rd}}$ of transformer blocks and a standard random initialization for the others.

Weight structure. The model with shuffled weights loses nearly all of the benefits of the procedural warm-up (Table 6). This confirms that the benefits are due to the model solving the pretraining objective and discovering the generating algorithm from the structure of the procedural data, rather than simply adjusting the magnitude of initial random weights.

Table 6. **Importance of weight structure.** Shuffling the weights after procedural warm-up and before image training, eliminates the performance gains, showing the benefits are due to precise weight structure rather than simply the distribution of their magnitudes.

Method	CIFAR-100 (%)
Random initialization	68.52
k DYCK	**71.98**
k DYCK, all weights shuffled	69.51

Component contributions. When shuffling weights within attention or MLP layers, the performance degrades substantially (Table 7). This means that the learned information is distributed throughout the network. The procedural warm-up establishes coherent relationships across layers that cannot be replicated with existing schemes like the Mimetic initialization that only target attention layers [51].

Layerwise analysis. Our selected ablation of early/mid/late layers shows that the procedural warm-up mostly affects the late layers, which account for most of the gains over the baseline (Table 8). This is a surprising observation because the pretraining of vision models normally relies mostly on early layers [27], which capture low-level transferable features. This suggests that the procedural warm-up provides a qualitatively different type of training signal than standard visual data.

Take-away. The useful information acquired during the procedural warm-up is encoded in *precise weight structure* of *both attention and MLP* components and concentrated almost exclusively in *late (deep) layers*.

Table 7. **Effect of component-wise weight shuffling.** Shuffling attention or MLP weights after procedural warm-up reduces downstream performance, showing that the inductive bias is *distributed throughout the network* rather than confined to a single component.

Method	**CIFAR-100** (%)
Random initialization	68.52
k DYCK	**71.98**
k DYCK, attention weights shuffled	70.57
k DYCK, MLP weights shuffled	70.71

Table 8. **Layerwise transfer.** Utilising only subsets of layers from the procedurally pretrained model shows that the late layers are the most useful. Early and middle layers provide limited benefit, suggesting that procedural warm-up imparts an inductive bias expressed in the network's deeper stages.

Transferred layers	**CIFAR-100** (%)
Random initialization	68.52
All Layers	**71.98**
First 4 layers	68.91
Middle 4 layers	70.19
Final 4 layers	<u>71.66</u>

5. Discussion

We proposed a *procedural warm-up* for ViTs, a generic strategy to better prepare ViT weights for subsequent learning of vision tasks. The method trains the model for masked-token prediction on sequences of abstract tokens generated with formal languages. These sequences have no semantic meaning or visual structure, but they force the model to learn simple algorithms, such as a stack mechanism with sequences of balanced parentheses. Remarkably, the ViTs trained on this data acquire soft inductive biases that help the subsequent training on image datasets. We observe improvements in the speed of convergence and final accuracy. Our experiments suggest that the procedural warm-up provides a training signal that is qualitatively different from natural images. This means that it does not only provide a head-start on standard training, e.g. on ImageNet. Instead, it provides information that would not be acquired otherwise.

Generality and efficiency. Our method is computationally lightweight since it uses only a small number of training steps and data that can be cheaply generated on the fly. Even more importantly, the method and the data are extremely generic. We observe benefits on a range of datasets with no particular tuning, and prior work [19, 22, 44] even shows benefits on natural language, computer code, and mathematics with a very similar approach. There is therefore reason to believe that the procedural warm-up instils a set of generic or universal computational mechanisms in the model. This means that a "warmed-up model" can be seen as an alternative to a standard random or structured initialization [20, 51, 58] since the computational cost can be amortised over many trained models.

An ambitious prospect is to develop a closed-form instantiation of procedurally-pretrained weights. Section 4.3 shows that precise structure in the weights is critical. Yet the Kolmogorov complexity of the model (i.e. its minimal description length) is necessarily small, since the data-generating algorithm fits in a few lines of code. Conceptually, it should therefore be possible to summarize a procedurally-pretrained model, perhaps in a simple formula like in the Mimetic initialization [51] of attention layers.

Our findings suggest that "learning to see" can be jumpstarted from procedural data, pointing toward a new path for data-efficient and modality-agnostic pretraining. They also raise broader questions about how inductive biases emerge and propagate within transformers: why do later layers absorb the greatest benefit from procedural warm-up, and how do non-visual priors interact with visual features during finetuning? Extending this analysis across architectures, modalities, and other forms of algorithmic data could reveal more general principles underlying transferable and universal representation learning.

Limitations. (1) We focus on standard ViTs, but future work could explore other architectures like Swin [31] and XCiT [12]. (2) We experimented with a limited set of procedural generators, with many possible alternatives [22, 55]. (3) The effects could be evaluated in other settings like segmentation, OOD generalization, and multimodal models.

References

[1] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023. 1

[2] Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. In *Advances in Neural Information Processing Systems*, 2021. 2

[3] Manel Baradad, Chun-Fu Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. Procedural image programs for representation learning. In *Advances in Neural Information Processing Systems*, 2022. 2, 3

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101: Mining discriminative components with random forests. In *Proceedings of the European Conference on Computer Vision*, 2014. 4

[5] Hao Chen, Abdul Waheed, Xiang Li, Yidong Wang, Jindong Wang, Bhiksha Raj, and Marah I Abdin. On the diversity of synthetic data and its impact on training large language models. *arXiv preprint arXiv:2410.15226*, 2024. 1

[6] Cheng-Han Chiang and Hung-Yi Lee. On the transferability of pre-trained language models: A study from artificial datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 3

[7] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 1959. 3

[8] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011. 4

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 4

[10] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 4, 5

[11] Junhao Du, Chuqin Zhou, Ning Cao, Gang Chen, Yunuo Chen, Zhengxue Cheng, Li Song, Guo Lu, and Wenjun Zhang. Large language model for lossless image compression with visual prompts. *arXiv preprint arXiv:2502.16163*, 2025. 2

[12] Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021. 8

[13] Adithya Giri. Ibit: Utilizing inductive biases to create a more data efficient attention mechanism. *arXiv preprint arXiv:2509.22719*, 2025. 2

[14] Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023. 1

[15] Michael Goodale, Salvador Mascarenhas, and Yair Lakretz. Meta-learning neural mechanisms rather than bayesian priors. *arXiv preprint arXiv:2503.16048*, 2025. 3

[16] Junlin Han, Shengbang Tong, David Fan, Yufan Ren, Koustuv Sinha, Philip Torr, and Filippos Kokkinos. Learning to see before seeing: Demystifying llm visual priors from language pre-training. *arXiv preprint arXiv:2509.26625*, 2025. 1, 2

[17] Zexue He, Graeme Blackwood, Rameswar Panda, Julian McAuley, and Rogerio Feris. Synthetic pre-training tasks for neural machine translation. In *Findings of the Association for Computational Linguistics*, 2023. 3

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 2

[19] 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*, 2025. 1, 2, 3, 5, 8

[20] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *Proceedings of the International Conference on Machine Learning*, 2020. 2, 5, 8

[21] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Proceedings of the International Conference on Machine Learning*, 2024. 1

[22] Liangze Jiang, Zachary Shinnick, Anton van den Hengel, Hemanth Saratchandran, and Damien Teney. Procedural pre-training: Warming up language models with abstract data. *arXiv preprint arXiv:2601.21725*, 2026. 1, 2, 3, 4, 5, 8

[23] 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. 2, 4, 5

[24] Hirokatsu Kataoka, Ryo Hayamizu, Ryosuke Yamada, Kodai Nakashima, Sora Takashima, Xinyu Zhang, Edgar Josafat Martinez-Noriega, Nakamasa Inoue, and Rio Yokota. Replacing labeled real-image datasets with auto-generated contours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009. 4

[26] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. Stanford University CS231N Course Project, 2015. 4

[27] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015. 2, 7

[28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 2

[29] Matthias Lindemann, Alexander Koller, and Ivan Titov. Sip: Injecting a structural inductive bias into a seq2seq model by simulation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. 3

[30] Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, et al. Best practices and lessons learned on synthetic data. *arXiv preprint arXiv:2404.07503*, 2024. 1

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 8

[32] R. Thomas McCoy and Thomas L. Griffiths. Modeling rapid language learning by distilling bayesian priors into artificial neural networks. *arXiv preprint arXiv:2305.14701*, 2023. 3

[33] Ryo Nakamura, Hirokatsu Kataoka, Sora Takashima, Edgar Josafat Martinez-Noriega, Rio Yokota, and Nakamasa Inoue. Pre-training vision transformers with very limited synthesized images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2

[34] Ryo Nakamura, Ryu Tadokoro, Ryosuke Yamada, Yuki M. Asano, Iro Laina, Christian Rupprecht, Nakamasa Inoue, Rio

Yokota, and Hirokatsu Kataoka. Scaling backwards: Minimal synthetic pre-training? In *Proceedings of the European Conference on Computer Vision*, 2024. 2

[35] Kodai Nakashima, Hirokatsu Kataoka, and Yutaka Satoh. Does formula-driven supervised learning work on small datasets? *IEEE Access*, 2023. 2

[36] Isabel Papadimitriou and Dan Jurafsky. Injecting structural hints: Using language models to study inductive biases in language learning. *arXiv preprint arXiv:2304.13060*, 2023. 3

[37] Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. How does code pretraining affect language model task performance? *arXiv preprint arXiv:2409.04556*, 2024. 1

[38] Zeyu Qin, Qingxiu Dong, Xingxing Zhang, Li Dong, Xiaolong Huang, Ziyi Yang, Mahmoud Khademi, Dongdong Zhang, Hany Hassan Awadalla, Yi R Fung, et al. Scaling laws of synthetic data for language models. *arXiv preprint arXiv:2503.19551*, 2025. 1

[39] Parsa Rahimi, Damien Teney, and Sebastien Marcel. Auggen: Synthetic augmentation can improve discriminative models. In *Advances in Neural Information Processing Systems*, 2025. 1, 2

[40] Ryokan Ri and Yoshimasa Tsuruoka. Pretraining with artificial language: Studying transferable knowledge in language models. *arXiv preprint arXiv:2203.10326*, 2022. 3

[41] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision*, 2016. 2

[42] Daniel L. Ruderman. Origins of scaling in natural images. *Vision Research*, 1997. 2

[43] Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Stephanie Fu, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, and Antonio Torralba. A vision check-up for language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1

[44] Zachary Shinnick, Liangze Jiang, Hemanth Saratchandran, Anton van den Hengel, and Damien Teney. Transformers pretrained on procedural data contain modular structures for algorithmic reasoning. In *ICML Workshop on Methods and Opportunities at Small Scale*, 2025. 1, 3, 8

[45] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5, 2018. 2

[46] Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025. 6

[47] Sora Takashima, Ryo Hayamizu, Nakamasa Inoue, Hirokatsu Kataoka, and Rio Yokota. Visual atoms: Pre-training vision transformers with sinusoidal waves. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[48] Damien Teney, Armand Mihai Nicolicioiu, Valentin Hartmann, and Ehsan Abbasnejad. Neural redshift: Random networks are not random functions. 2024. 1

[49] Shengbang Tong, David Fan, Jiachen Li, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17001–17012, 2025. 2

[50] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 2

[51] Asher Trockman and J. Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023. 2, 4, 5, 7, 8

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

[53] Yanwei Wang, Ching-Yun Ko, and Pulkit Agrawal. Visual pre-training for navigation: What can we learn from noise? *arXiv preprint arXiv:2207.00052*, 2022. 3

[54] Yuhuai Wu, Markus N. Rabe, Wenda Li, Jimmy Ba, Roger B. Grosse, and Christian Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning. In *Proceedings of the International Conference on Machine Learning*, 2021. 3

[55] Yuhuai Wu, Felix Li, and Percy S. Liang. Insights into pretraining via simpler synthetic tasks. *Advances in Neural Information Processing Systems*, 2022. 1, 3, 8

[56] Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones. *arXiv preprint arXiv:2311.18823*, 2023. 11

[57] Shiyang Zhang, Aakash Patel, Syed A. Rizvi, Nianchen Liu, Sizhuang He, Amin Karbasi, Emanuele Zappala, and David van Dijk. Intelligence at the edge of chaos. *arXiv preprint arXiv:2410.02536*, 2024. 1, 3

[58] Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *arXiv preprint arXiv:2110.12661*, 2021. 2, 5, 8

[59] Boyang Zheng, Jinjin Gu, Shijun Li, and Chao Dong. Lm4lv: A frozen large language model for low-level vision tasks. *arXiv preprint arXiv:2405.15734*, 2024. 2

[60] Jianqiao Zheng, Xueqian Li, Hemanth Saratchandran, and Simon Lucey. Structured initialization for vision transformers. *arXiv preprint arXiv:2505.19985*, 2025. 2