WINR: NEURAL IMPLICIT REPRESENTATIONS FOR WEIGHT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Training large neural networks from scratch for every architectural variant remains a major computational bottleneck. Model growth methods reduce this cost by warm-starting larger target networks from smaller pretrained checkpoints, but they typically rely on hand-crafted rules such as weight tiling or layer duplication. These discrete heuristics restrict expansion to rigid scaling patterns and initialize large portions of the target network as direct copies of the source. In this paper, we ask whether model growth can instead be formulated as querying a learned continuous weight field, and propose WINR. Rather than treating pretrained parameters as arrays to be copied, WINR views them as samples of an underlying function that maps structured geometric and semantic coordinates to weights. A single field, fitted to one or more pretrained networks, can be queried to synthesize weights for target architectures with new shapes, depths, or layer compositions. This formulation naturally supports fractional expansion and cross-architecture synthesis beyond standard copy-based growth rules. Experiments on transformer models below 400M parameters show that WINR improves convergence over random initialization at fixed token budgets, and that a field fitted to multiple source architectures is competitive with existing model growth methods, with performance improving as source diversity increases.

1 Introduction

Training large neural networks from scratch for every architectural variant is a major computational bottleneck. Pre-training Llama-3-8B, for instance, required approximately 7.7M GPU-hours, and changing basic choices such as hidden size, depth, or attention head count often means repeating this process (Kaplan et al., 2020; Hoffmann et al., 2022; Dubey et al., 2024). Model growth methods reduce this cost by *warm-starting* larger targets from smaller pre-trained checkpoints (Chen et al., 2016). HyperCloning (Samragh et al., 2024) tiles a source matrix W_S into a $2\times$ or $4\times$ wider matrix W_D while preserving functionality, and G_{stack} (Du et al., 2024) duplicates a model's layers g times to initialize a $g\times$ deeper network. These methods show that pre-trained weights contain reusable structure that accelerates the training of larger models.

These methods, however, share a restrictive template: target parameters are populated by fixed, hand-crafted rules applied to discrete source tensors. Large fractions of the target are therefore exact copies of the source, which induces repeated structure in the initialization and restricts transfer to targets compatible with the growth rule. The underlying limitation is that weights are treated as values bound to a fixed discrete grid. This raises a question: *can model growth be formulated as querying a learned representation of pre-trained weights at new architectural coordinates, rather than copying or duplicating source tensors?*

We answer this by treating pre-trained weights as samples of an underlying continuous function. We introduce a *learned continuous weight field*, a mapping from parameter *coordinates* to parameter *values*, carrying the idea behind neural implicit representations such as NeRF and SIREN (Mildenhall et al., 2020; Sitzmann et al., 2020; Tancik et al., 2020) from signal space into weight space. Each scalar weight is indexed by a coordinate $\boldsymbol{x} = (d, r, c, m, \tau, a)$, where $(d, r, c) \in [0, 1]^3$ encode normalized depth and within-tensor row and column, and (m, τ, a) identify modality, layer role, and architecture family. We fit $f_\theta(\boldsymbol{x}) \approx w$ over the resulting $(\boldsymbol{x}, w)$ pairs (Figure 1) and synthesize a target model by querying f_θ on the coordinate grid of the target architecture. Parameter synthesis is thus decoupled from the tensor dimensions of any particular source checkpoint.

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.

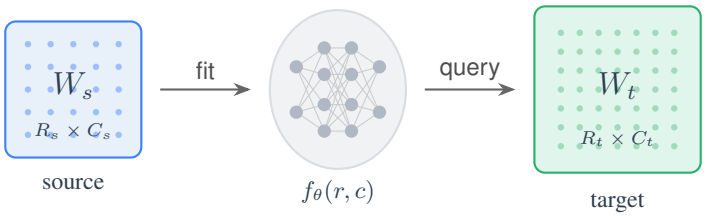


Figure 1: A trained INR $f_\theta(r, c)$ expands a source tensor $W_s \in \mathbb{R}^{R_s \times C_s}$ into a target tensor $W_t \in \mathbb{R}^{R_t \times C_t}$ of arbitrary shape by fitting a continuous field to the source weights and querying it at the target coordinates. For clarity, this illustration uses only row and column coordinates to demonstrate tensor expansion.

This view addresses two limitations of copy-based growth. First, querying f_θ on any $R' \times C'$ grid yields a tensor of that shape, enabling fractional expansion without a predefined tiling rule. Second, fitting shared fields across several pre-trained models amortizes them over a model zoo, and structural conditioning shares regularities across layer types and architecture families. We study two regimes: a *single-architecture* setting, where the field is fitted to one model and used to synthesize variants of the same architecture, and a *multi-architecture* setting, where shared fields are fitted across a heterogeneous zoo and queried for held-out tensor shapes and unseen architectural configurations.

Contributions.

1. We introduce WINR, a continuous neural-field formulation for model initialization that replaces rule-based tensor expansion with coordinate-based parameter synthesis.
2. We develop a unified coordinate representation and a scalable generator architecture that model heterogeneous parameter tensors across layers and architectures in a shared continuous space.
3. We show that this formulation supports fractional expansion and cross-architecture synthesis, which copy-based growth rules do not.
4. We evaluate WINR on transformer models below 400M parameters, showing high-fidelity checkpoint reconstruction, faster convergence or better final performance than random initialization at fixed token budgets, and, with multiple source architectures, performance competitive with established model-growth methods.

2 Related Work

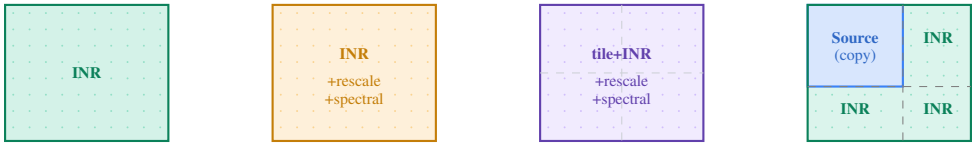
Model growth for efficient pre-training. Net2Net (Chen et al., 2016) introduced function-preserving width/depth expansion via neuron splitting and identity-layer injection, later extended to BERT by Bert2Bert (Chen et al., 2022). More recent LLM-scale methods include StackedBert's depthwise duplication (Gong et al., 2019), LiGO's learned matrix mapping (Wang et al., 2023), MSG's masked random growth (Yao et al., 2024), Lemon (Wang et al., 2024b), and FLM-101B (Li et al., 2025). Two stand out as strong baselines. HyperCloning (Samragh et al., 2024) achieves 2.2–4 $\times$ speedup on OPT, Pythia, and OLMO by function-preserving width cloning. G_{stack} (Du et al., 2024) systematically compares four atomic growth operators and finds that direct depthwise stacking yields 54.6% speedup on a 7B model trained with 300B tokens, with an empirical scaling law relating growth timing d and growth factor g to target FLOPs.

Neural implicit representations. Our coordinate formulation is a direct conceptual descendant of NeRF (Mildenhall et al., 2020), SIREN (Sitzmann et al., 2020), and random Fourier features (Tancik et al., 2020; Rahimi & Recht, 2007), which represent images, 3D scenes, and signed distance fields as continuous coordinate-to-value functions. We transplant this substrate from perceptual signals to the *parameter tensors themselves*.

Weight-space learning and hypernetworks. Hypernetworks (Ha et al., 2017) generate the weights of a target network conditioned on a task or latent code. Follow-up work generates parameters from

Table 1: Positioning against prior model-growth and weight-generation methods.

Method	Target dim.	Multi-source	Cross-arch.	Func. preserving
HyperCloning (Samragh et al., 2024)	integer ×	no	no	yes
G_{stack} (Du et al., 2024)	integer ×	no	no	no
LiGO (Wang et al., 2023)	integer ×	no	no	no
Hypernetworks (Ha et al., 2017)	fixed	via cond.	limited	—
WINR	**arbitrary**	**yes**	**yes**	no (by design)



(a) `inr` (b) `inr_rescale` (c) `inr_blend` (d) `inr_extend`

Figure 2: **Expansion modes.** Each panel shows the target tensor $R_t \times C_t$. **(a)** Pure field evaluation. **(b)** Field + fan-in rescale + spectral restore. **(c)** Tiled source blended with field, then corrections. **(d)** Pre-trained source in top-left block (blue); field fills the rest (green).

model metadata, treats weights as a learnable distribution (Schürholt et al., 2022; Peebles et al., 2022), or diffuses in weight space (Peebles et al., 2022; Wang et al., 2024a). WINR differs in that it has *no global latent code*: generation is a direct coordinate lookup, which is what permits dimension-agnostic querying.

3 Method

We reconceptualize pre-trained parameter tensors as discrete evaluations of an underlying continuous weight field over a structured coordinate space. We introduce this framework by first examining a single-parameter tensor, which naturally motivates a tensor-specific implicit neural representation (INR) mapping geometric coordinates to scalar weight values. Having established this per-parameter foundation, we then generalize the construction to a shared, conditional INR capable of modeling a vast population of tensors across diverse pre-trained architectures. By conditioning on both geometric position and semantic topology (e.g., layer type, modality, architecture family), we transform the discrete, high-dimensional parameter space into a continuous, optimizable manifold. Consequently, evaluating this learned function at unseen coordinates will enable seamless structural expansion and smooth interpolation directly within a smooth neural function space.

3.1 Coordinate representation

To model a diverse population of neural networks within a single continuous space, we define a unified coordinate system that captures both the geometric and semantic properties of every parameter.

We first homogenize all parameter topologies into 2D matrices $T \in \mathbb{R}^{R\times C}$ to establish a uniform geometric foundation. For linear layers with weights $W \in \mathbb{R}^{O\times I}$ and biases $b \in \mathbb{R}^{O}$, we construct the augmented matrix $T = [W \mid b] \in \mathbb{R}^{O\times(I+1)}$. Normalization parameters are similarly stacked as $T = [w \mid b] \in \mathbb{R}^{D\times 2}$. Each scalar entry $T_{i,j}$ is then assigned normalized spatial coordinates $(r_i, c_j) = (\frac{i}{R-1}, \frac{j}{C-1}) \in [0,1]^2$. To situate these matrices within the macroscopic network hierarchy, we append a depth coordinate $d \in [0,1]$. For an L-layer architecture, we anchor the extremes by setting $d = 0$ for input embeddings and $d = 1$ for output heads, while uniformly spacing intermediate blocks $\ell \in \{0, \dots, L-1\}$ at $d = (\ell+1)/(L+2)$.

Because geometric coordinates alone cannot isolate functionally orthogonal components (e.g., attention projections vs. MLP weights) that share identical spatial positions, we introduce discrete semantic identifiers. Variables for modality m, layer type τ, and architecture family a are mapped to learnable embeddings (e_m, e_τ, e_a).

By aggregating every scalar weight across a zoo of pre-trained models into a composite coordinate $\boldsymbol{x} = (d, r, c, e_m, e_\tau, e_a)$ paired with its ground-truth scalar value $w = T_{i,j}$, we form a dense

coordinate-to-weight dataset $\mathcal{D} = \{(\boldsymbol{x}_n, w_n)\}$. We train a single, shared conditional implicit neural representation (INR), f_θ, via a global reconstruction objective:

$$\min_\theta \mathbb{E}_{(\boldsymbol{x},w)\sim\mathcal{D}} \left[\|f_\theta(\boldsymbol{x}) - w\|_2^2 \right]. \tag{1}$$

Once optimized, f_θ encapsulates the continuous parameter manifold for the entire model zoo. To instantiate a novel target architecture, we do not copy discrete blocks; instead, we systematically generate a new coordinate grid that reflects the target's desired layer dimensions and depth. Evaluating f_θ at these novel, arbitrary coordinates natively generates the requisite weight matrices, directly facilitating zero-shot structural resizing, fractional depth scaling, and cross-architecture weight interpolation.

3.2 Generator and training objective

To parameterize the continuous weight field f_θ, we employ a coordinate-conditioned residual network. Because standard MLPs exhibit a spectral bias against high-frequency detail in low-dimensional spaces (Tancik et al., 2020), we project the geometric coordinates $\mathbf{g} = (d, r, c) \in [0, 1]^3$ using a *dual-band random Fourier feature* encoding. Let $\mathbf{B}_\ell, \mathbf{B}_h \in \mathbb{R}^{3\times M}$ be fixed random matrices with entries drawn i.i.d. from $\mathcal{N}(0, 1)$, scaled by σ_ℓ and σ_h. The Fourier map is defined as:

$$\gamma(\mathbf{g}) = \left[\sin(2\pi\mathbf{g}\mathbf{B}_\ell),\ \cos(2\pi\mathbf{g}\mathbf{B}_\ell),\ \sin(2\pi\mathbf{g}\mathbf{B}_h),\ \cos(2\pi\mathbf{g}\mathbf{B}_h) \right] \in \mathbb{R}^{4M}. \tag{2}$$

The low-frequency bank (σ_ℓ=4) captures coarse structural topology, while the high-frequency bank (σ_h=32) resolves the dense, fine-grained clusters characteristic of trained neural network weights. Both matrices are deterministic after initialization. The semantic variables (m, τ, a) are mapped to learned embeddings $\mathbf{e}_m, \mathbf{e}_\tau, \mathbf{e}_a \in \mathbb{R}^E$, yielding the composite input vector $\mathbf{z} = [\gamma(\mathbf{g}) \parallel \mathbf{e}_m \parallel \mathbf{e}_\tau \parallel \mathbf{e}_a] \in \mathbb{R}^{4M+3E}$. In the single-architecture regime, semantic embeddings are omitted, collapsing $\mathbf{z}$ to purely $\gamma(\mathbf{g})$.

We process this composite representation $\mathbf{z}$ through a residual MLP to parameterize the field f_θ. An input projection $\mathbf{h}_0 = \sigma(\mathbf{W}_0\mathbf{z} + \mathbf{b}_0)$ maps $\mathbf{z}$ to a D-dimensional hidden space, utilizing GELU activations (Hendrycks & Gimpel, 2016). This is followed by B residual blocks:

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \mathbf{W}_{i,2}\sigma(\mathbf{W}_{i,1}\mathbf{h}_i + \mathbf{b}_{i,1}) + \mathbf{b}_{i,2}, \qquad i = 0, \ldots, B-1, \tag{3}$$

terminating in a scalar linear head $\hat{w} = \mathbf{w}_{\text{out}}^\top \sigma(\mathbf{h}_B) + b_{\text{out}}$. To stabilize early training, we apply scaled Xavier uniform initialization (gain 0.5 for residual branches, 0.1 for the output head).

Fitting this architecture to pre-trained weights presents a unique optimization challenge. Pre-trained weight distributions are sharply concentrated near zero but exhibit extremely heavy tails, rendering direct ℓ_2 regression numerically unstable. We mitigate this via an *arcsinh normalization* that logarithmically compresses extreme values while preserving signs and near-zero linearity. Given a tensor $\mathbf{T}$, let $s = \max(Q_{75}(|\mathbf{T}|), 10^{-8})$ denote its robust scale. The normalized target is:

$$\tilde{T}_{ij} = \operatorname{arcsinh}\Big(\frac{T_{ij}}{s}\Big) = \ln\Big(\frac{T_{ij}}{s} + \sqrt{\frac{T_{ij}^2}{s^2} + 1}\Big). \tag{4}$$

This monotone bijection maps the sparse, heavy-tailed distribution to a well-behaved, near-Gaussian range. At inference, the exact original scale is recovered via the inverse transform $\hat{T}_{ij} = s \cdot \sinh(\hat{\tilde{T}}_{ij})$.

To complement this normalized space, we optimize f_θ using a log-cosh loss over the targets: $\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \log\cosh(\hat{\tilde{w}}_n - \tilde{w}_n)$, implemented via the numerically stable form $x + \text{softplus}(-2x) - \log 2$. The log-cosh objective acts as $\frac{1}{2}\Delta^2$ for small residuals, ensuring high-fidelity reconstruction of the dense near-zero cluster, while smoothly transitioning to an ℓ_1-like penalty ($|\Delta| - \log 2$) for large errors. This circumvents the gradient-explosion issues of pure ℓ_2 loss on outliers and the origin-bias of pure ℓ_1 loss. Full optimization details, including hyperparameters and training schedules, are provided in Appendix A.2.

3.3 Cross-Architecture Generalization via Graph-Conditioned Generation

The conditional INR of Section 3.2 treats each architecture as an opaque categorical token, discarding the structural relationships—head counts, key-value grouping, gated activations, expansion

ratios—that govern weight geometry. To generalize f_θ across a heterogeneous zoo, we replace the flat semantic embeddings with a *topology-aware* conditioning mechanism derived from each architecture's computational graph.

We represent each model as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mirroring its forward-pass dataflow, where nodes correspond to functional components and edges encode computational dependencies. Each node v carries a feature vector combining one-hot type and family indicators, normalized structural scalars (hidden dimension, head counts, expansion ratio, etc.), and log-scaled geometric descriptors (Appendix B). A lightweight GNN processes $\mathcal{G}$ through $K{=}3$ rounds of residual message passing:

$$\mathbf{h}_v^{(k+1)} = \mathbf{h}_v^{(k)} + \phi^{(k)}\Big(\big[\, \mathbf{h}_v^{(k)},\ \tfrac{1}{|\mathcal{N}(v)|} \textstyle\sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k)} \,\big] \Big), \tag{5}$$

yielding a global architecture embedding $\mathbf{z}_{\text{arch}}$ via mean-pooling and per-node embeddings $\mathbf{z}_v$ via a separate projection head, both trained jointly with the INR.

For each scalar weight at node v and relative depth δ, we assemble $\mathbf{c} = [\, \delta,\ \mathbf{z}_v,\ \mathbf{z}_{\text{arch}} \,]$ and inject it into the generator via FiLM modulation (Perez et al., 2017), replacing the residual blocks of Eq. 3 with:

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \boldsymbol{\gamma}_i(\mathbf{c}) \odot \sigma\big(\mathbf{W}_{i,1}\mathbf{h}_i + \mathbf{b}_{i,1}\big) + \boldsymbol{\beta}_i(\mathbf{c}), \tag{6}$$

where $\boldsymbol{\gamma}_i, \boldsymbol{\beta}_i$ are initialized to identity scaling and zero shift for stability. We train one such graph-conditioned INR per functional parameter type τ (e.g., `attn_q`, `mlp_down`), pooling all matrices of a given type across every architecture in the zoo into a single globally shuffled dataset per epoch. This flat-pool strategy, combined with per-batch graph encoding amortization, ensures the INR learns architecture-invariant spatial structure modulated exclusively through the graph-derived conditioning pathway, enabling generalization to unseen model families at inference time. Full construction and training details are provided in Appendix B.

3.4 Expansion via Coordinate Resampling

By encoding the source model as a continuous weight field f_θ, we unify reconstruction and expansion into a single operation. Because f_θ parameterizes an underlying function rather than a rigid matrix, evaluating it at the original training coordinates regenerates the source, while evaluating the *exact same function* over a denser coordinate grid natively instantiates an expanded target architecture. To synthesize a target matrix $\mathbf{W}_t \in \mathbb{R}^{R_t \times C_t}$ (where $R_t \geq R_s$ or $C_t \geq C_s$), we directly query this continuous source representation over normalized grid coordinates and invert the robust scaling: $\hat{w}_{i',j'} = s \cdot \sinh(f_\theta(i'/(R_t-1), j'/(C_t-1)))$. We map these continuous evaluations into functional network weights via four distinct expansion strategies (Figure 2).

For pure continuous synthesis, the `inr` mode evaluates the source field over the novel target grid without post-processing. However, because densely sampling a bandlimited field inherently induces rank collapse, the `inr_rescale` mode mathematically corrects this by applying fan-in variance scaling, $\mathbf{W}_t \leftarrow \mu + (\mathbf{W}_t - \mu)\sqrt{C_s/C_t}$, and injecting Marchenko–Pastur-distributed noise into the null space to restore spectral rank (Appendices C.1, C.2).

Alternatively, hybrid strategies blend this continuous source representation with the original discrete checkpoint to strongly anchor the target optimization space. The `inr_blend` mode linearly interpolates between a discretely tiled source matrix and the continuous INR evaluation, explicitly combining exact coarse structure with learned high-frequency detail. Conversely, the `inr_extend` mode strictly preserves the exact source weights within the upper-left principal submatrix ($\mathbf{W}_t[:R_s, :C_s] = \mathbf{W}_s$), querying f_θ exclusively to populate the newly allocated peripheral dimensions (Figure 2d).

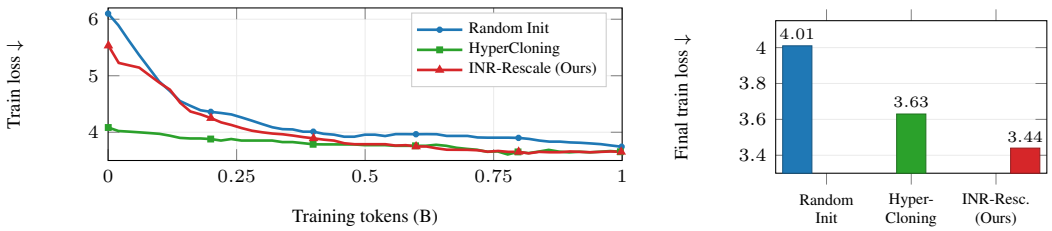
Finally, normalization parameters and embeddings are synthesized via dedicated INRs under analogous logic (Appendix C.5). To guarantee input-output dimensional consistency, the target output head is explicitly tied to the expanded token embedding matrix.

4 Experiments

We evaluate WINR in three settings of increasing scope: single-architecture reconstruction and width expansion of MobileLLM-R1-140M-base to MobileLLM-R1-360M (Section 4.1); cross-architecture synthesis of Gemma-3-270M from a field fitted to MobileLLM (Changsheng et al.,

Table 2: Functional preservation under single-architecture INR fitting. WINR is trained to reconstruct the parameters of MobileLLM-R1-140M-base and evaluated against the original checkpoint across mathematical reasoning, code generation, and commonsense benchmarks.

Model	Size	MATH500 4-shot EM	GSM8K 8-shot EM	MBPP 3-shot pass@1	HumanEval 0-shot pass@1	CommonSense 0-shot avg. acc.
SmolLM2-135M	135M	0.4	1.8	3.8	0.0	50.7
MobileLLM-R1-140M-base	140M	4.6	**16.3**	5.4	15.9	44.3
WINR	140M	**4.8**	16.2	**5.6**	**16.1**	44.7



(a) Training loss vs. tokens.

(b) Final loss at 1.0 B tokens.

Figure 3: Training dynamics for MobileLLM-360M on SlimPajama. **(a)** Loss curves over 1 B tokens. **(b)** Final loss. INR-Rescale reduces loss by 14% vs. random init and 5% vs. HyperCloning.

2025) and SmolLM (Allal et al., 2025) checkpoints (Section 4.2); and multi-source BERT fields decoded into BERT-12L-768H (Section 4.3). MobileLLM and Gemma are pre-trained on a subset of SlimPajama (Soboleva et al., 2023), and BERT on English WikiText.

We report perplexity on WikiText (Merity et al., 2016), Text8 (Al-Rfou et al., 2018), and LAMBADA (Paperno et al., 2016), and zero-shot accuracy on MMLU, HellaSwag, ARC-Easy, ARC-Challenge, PIQA, and WinoGrande. BERT models are evaluated by pre-training loss and GLUE fine-tuning. We compare against random initialization in the growth settings. For MobileLLM, whose source and target dimensions are not related by a uniform integer factor, we add a generalized HyperCloning-style tiling baseline; for BERT, we compare against LEMON (**?**) and training from scratch. Existing growth baselines define no cross-family transformation for Gemma, so we compare it against random initialization. MobileLLM uses `inr_rescale`, while Gemma and BERT use pure field decoding without source tiling or blending (modes in Table 5). Unless stated otherwise, means and standard deviations are over three seeds (42, 1234, 7777); the 1B-token MobileLLM runs, the per-step Gemma trajectories, and the one- to three-source BERT fields are single runs.

4.1 Single-Architecture Reconstruction and Width Expansion

We first ask whether a continuous field can faithfully represent a pre-trained checkpoint. We fit WINR to MobileLLM-R1-140M-base using a separate INR for each attention and MLP projection with approximately 50% of the tensor's parameter count (2× compression), a shared INR for the concatenated RMSNorm scale vectors, and chunk-conditioned INRs for the embedding matrix. The reconstructed checkpoint closely matches the original across mathematical reasoning, code generation, and commonsense benchmarks (Table 2).

Width expansion. We expand MobileLLM-R1-140M-base to MobileLLM-R1-360M at a fixed depth of 15 layers using `inr_rescale`, which combines field evaluation with fan-in rescaling and spectral restoration. At a budget of 1B tokens, WINR reaches lower WikiText perplexity than random initialization throughout training and a final training loss of 3.44, compared with 3.63 for tiling and 4.01 for random initialization (Figures 3a and 3b). Tiling starts from a lower loss but plateaus earlier. The learned field therefore provides a useful warm start even when the target width cannot be obtained through integer-factor expansion.

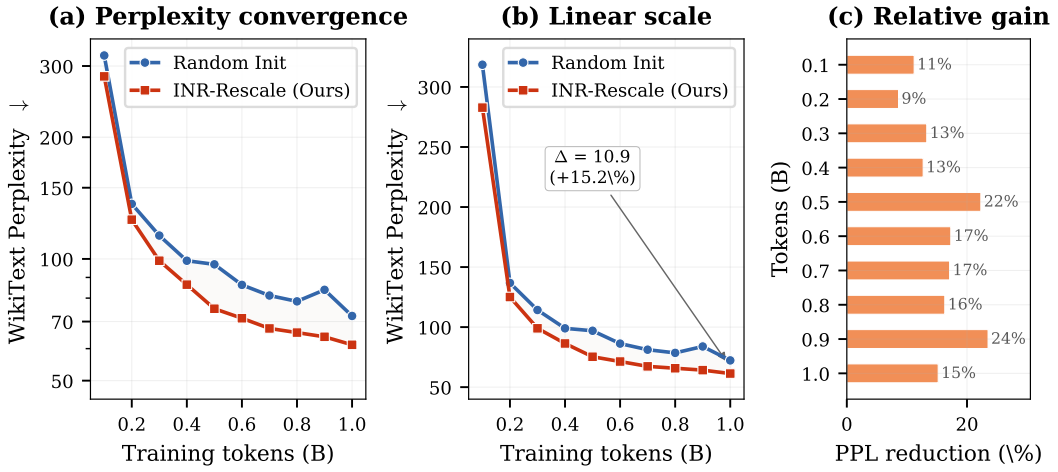


Figure 4: **WikiText perplexity convergence under training on Slimpajama dataset.** We compare random initialization with INR-RESCALE initialization under the same training-token setting. INR-RESCALE consistently reaches lower WikiText perplexity throughout training, indicating faster convergence and better sample efficiency. The linear-scale view highlights an average perplexity reduction of $\Delta = 10.9$ points, corresponding to a 15.2% relative improvement, while the right panel reports token-wise relative gains, reaching up to 24%.

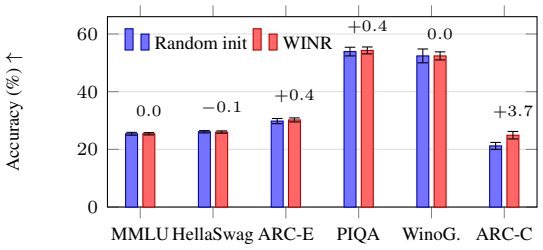


Figure 5: **Gemma-3-270M accuracy at 15K steps**, mean $\pm$ std over three seeds. Labels: WINR minus random init (pp).

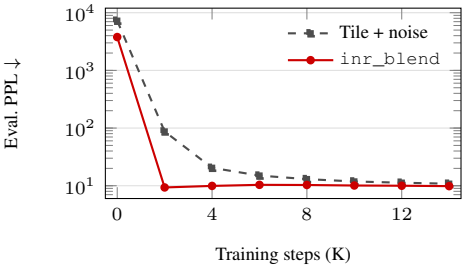


Figure 6: **Tile-plus-noise control**, BERT-6L-384H $\rightarrow$ 6L-512H, $\beta = 0.5$. Std ≤ 0.4.

4.2 Cross-Architecture Synthesis on Gemma-3-270M

We next test whether a field trained across multiple source architectures can initialize a structurally unseen target. We decode it into Gemma-3-270M, whose GQA ratio (2:1), attention pattern, vocabulary size (256K), and tokenizer are all absent from field training, and pre-train for 15K steps. Across three seeds (Figure 5), WINR is comparable to random initialization on MMLU, HellaSwag, ARC-Easy, PIQA, and WinoGrande, with differences of at most 0.4 pp and within the reported standard deviations. ARC-Challenge shows the largest mean difference, 24.9 ± 1.3 vs. 21.2 ± 1.2.

In the single run with recorded trajectories (Appendix, Figure 9), perplexities are comparable at 3K steps and lower for WINR from 9K onward, reaching 194.9 vs. 198.0 on WikiText, 571.3 vs. 572.3 on Text8, and 334.2 vs. 344.3 on LAMBADA at 15K. The field thus produces a trainable initialization for an unseen architecture, although gains over random initialization remain modest.

4.3 Multi-Source Fields and Comparison with LEMON

We study how source diversity affects pure field initialization by fitting WINR to one, two, three, and five BERT source architectures and decoding each field into BERT-12L-768H. The one-source field uses BERT-6L-384H, the two-source field adds BERT-6L-512H, and the five-source field uses five distinct architectures: BERT-6L-384H, BERT-6L-512H, BERT-12L-512H, BERT-12L-256H, and BERT-8L-512H. This setting also exercises linear biases and LayerNorm scale and bias parameters.

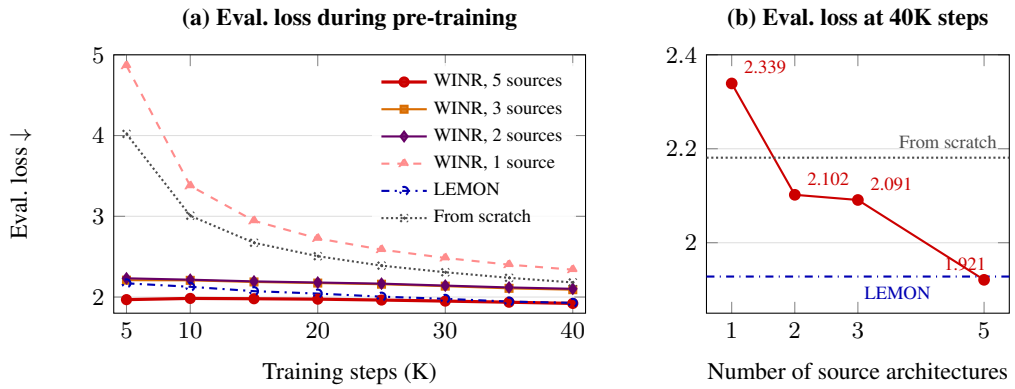


Figure 7: **Pure-field BERT growth to BERT-12L-768H.** WINR fields are fitted to 1, 2, 3, or 5 BERT source architectures and decoded without source tiling or blending. (a) Evaluation loss from 5K to 40K steps; step-0 losses are given in the text. (b) Evaluation loss at 40K steps against the number of sources, with training from scratch and LEMON as references. WINR with 5 sources, LEMON, and from scratch are means over three seeds; the 1, 2, and 3 source fields are single runs.

Table 3: **GLUE fine-tuning (3 epochs per task).** STS-B: Spearman correlation; CoLA: Matthews correlation; others: accuracy. All values $\times 100$. Best per column in bold within each block. (a) BERT-12L-768H targets from Section 4.3; WINR is the pure field fitted to five sources. (b) BERT-6L-512H targets from the tile-plus-noise control.

Method	STS-B	MRPC	CoLA	SST-2	QNLI	MNLI	MNLI-mm	QQP	Avg
(a) BERT-12L-768H									
From scratch	85.70±0.18	79.90±0.09	33.60±0.12	89.22±0.02	87.06±0.10	79.54±0.07	80.39±0.03	89.76±0.23	78.15±0.04
LEMON	**87.30**±0.01	**83.33**±0.17	24.80±0.09	90.37±0.10	**89.42**±0.05	**80.91**±0.02	81.46±0.01	**90.38**±0.08	78.50±0.03
WINR (5 sources)	85.40±0.66	82.11±0.27	**37.00**±0.37	**91.06**±0.09	88.93±0.33	80.66±0.90	**81.62**±0.03	90.23±0.13	**79.63**±0.16
(b) BERT-6L-512H									
Tile + noise	**85.40**±0.08	**80.64**±0.12	32.60±0.06	88.19±0.12	87.37±0.16	78.98±0.02	**79.57**±0.10	89.28±0.07	77.75±0.04
`inr_blend` ($\beta = 0.5$)	84.90±0.08	79.90±0.13	**38.20**±0.09	**89.68**±0.23	**87.86**±0.13	**79.18**±0.04	79.50±0.01	**89.31**±0.11	**78.57**±0.04

The one-source field is worse than training from scratch at every logged step, reaching 2.339 vs. 2.181 at 40K steps (Figure 7). Adding sources improves performance substantially: the two-, three-, and five-source fields reach 2.102, 2.091, and 1.921, compared with 1.928 for LEMON. Initial loss decreases from 10.326 with one source to 6.188 with five, compared with 1.996 for LEMON. The five-source field is below LEMON at every logged step from 5K to 40K; the gap is largest at 5K (1.968 vs. 2.170) and narrows to 0.007 by 40K. After GLUE fine-tuning (Table 3(a)), the five-source field averages 79.63, compared with 78.50 for LEMON and 78.15 for training from scratch, and is higher than LEMON on CoLA, SST-2, and MNLI-mm but lower on STS-B, MRPC, QNLI, MNLI, and QQP.

Pure field initialization therefore benefits strongly from source diversity: a single-source field underperforms training from scratch, while more sources progressively improve optimization. We interpret this as improved early sample efficiency with final performance competitive with LEMON, rather than evidence of a better converged solution.

Table 4: **Expansion strategy ablation**, MobileLLM-R1-140M-base → MobileLLM-R1-360M. WikiText-2 PPL (↓), mean ± std over three seeds. Best in bold, second best underlined. Δ: reduction vs. `inr` at 45M.

Strategy	5M	25M	45M	Δ(%)
`inr`	3325 ± 4.4	1961 ± 3.1	1730 ± 2.7	–
`inr_extend`	3176 ± 2.1	1940 ± 3.1	1685 ± 6.2	2.6
`inr_src`	3183 ± 1.1	1920 ± 3.4	1664 ± 3.4	3.8
`inr_rescale`	3167 ± 0.4	1890 ± 5.3	1645 ± 4.4	4.9
`inr_merge`	3117 ± 2.4	<u>1764</u> ± 2.1	<u>1526</u> ± 0.1	<u>11.8</u>
`inr_tile_merge`	<u>3108</u> ± 0.2	<u>1764</u> ± 2.1	<u>1526</u> ± 1.1	<u>11.8</u>
`inr_blend`	**3033** ± 4.2	**1693** ± 1.9	**1439** ± 0.4	**16.8**

4.4 Ablations

Expansion strategy. We compare expansion modes for MobileLLM-R1-140M-base → MobileLLM-R1-360M, training each for 45M tokens on WikiText-2 (Table 4). Pure field decoding (`inr`) is weakest (1730 ± 2.7 perplexity), and `inr_rescale` improves it to 1645±4.4. Source-anchored modes perform better: `inr_blend` is best at every budget (1439 ± 0.4, a 16.8% reduction over `inr`), and its gain over `inr_merge` (1526) isolates the effect of fan-in rescaling and spectral restoration. Replacing the exact source tensor

with its field reconstruction has little effect: `inr_tile_merge` matches `inr_merge` (1526 ± 1.1 vs. 1526 ± 0.1), and `inr_src` reaches 1664 ± 3.4 vs. 1685 ± 6.2 for `inr_extend`. In the single-source setting, pure field decoding is thus weaker than source-anchored variants, while the field reconstruction can largely substitute for exact source values within them.

Tile-plus-noise control. To test whether the field component of `inr_blend` can be replaced by an unstructured perturbation, we expand BERT-6L-384H to BERT-6L-512H with `inr_blend` ($\beta = 0.5$) and compare against tiling plus Gaussian noise matched in magnitude to the field component. `inr_blend` reaches perplexity 9.336 vs. 87.219 at 2K steps and remains lower through 14K steps (Figure 6); the two are comparable at 40K (8.900 vs. 8.841), and at the final 132K step `inr_blend` reaches 7.084 vs. 7.148. After GLUE fine-tuning (Table 3(b)), `inr_blend` is higher on five of eight tasks and on average (78.57 vs. 77.75). Matched-magnitude noise therefore does not reproduce the early optimization advantage of the coordinate-aligned field. The control at the smaller $\beta \in [0.01, 0.1]$ used for MobileLLM remains to be evaluated.

Field capacity and fitting cost. Larger fields fit the same weights substantially faster (Figure 8; full sweep in Appendix, Figure 11). The $1\times$, $2\times$, and $4\times$ fields reach reconstruction cosine 0.98 in 180.3 s, 62.9 s, and 50.4 s, while the $0.5\times$ field does not reach it within 2000 epochs. At cosine 0.99, only the $2\times$ and $4\times$ fields reach the threshold (187.5 s and 149.7 s). Field capacity is therefore a trade-off between compression and fitting efficiency rather than a fixed property of the method.

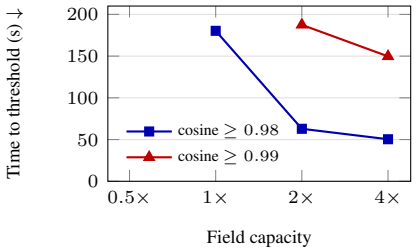


Figure 8: **Fitting time vs. field capacity.** Wall-clock from epoch 0 until reconstruction cosine first reaches each threshold. $0.5\times$ and $0.9\times$ reach neither threshold; $1\times$ does not reach 0.99. Full sweep: Figure 11.

5 Conclusion

We introduced WINR, a continuous neural-field framework for replacing discrete, rule-based tensor expansion with coordinate-based parameter synthesis. By fitting a field over pre-trained model weights and conditioning it on architectural structure, WINR decouples weight generation from the fixed geometry of any single source model, enabling fractional expansion, cross-architecture synthesis, and zoo-level amortization within a unified formulation. Empirically, WINR reconstructs pre-trained checkpoints with high fidelity, improves convergence over random initialization and copy-based baselines, and shows promising generalization to held-out architecture families. These results suggest that continuous weight fields provide a viable foundation for scalable, learned model expansion beyond hand-crafted growth rules.

Limitations.

Our evaluation is restricted to models below 400M parameters, and cross-architecture generation remains challenging when source and target models differ substantially in hidden dimension. In the single-source setting, pure field decoding underperforms source-anchored variants and, for BERT, training from scratch. The gains over random initialization on the held-out Gemma architecture are modest on most benchmarks, and end-to-end compute savings are not yet established. Scaling to larger models and improving cross-architecture transfer remain important directions for future work.

LLM Usage Disclosure.

We used large language models only as a writing assistance tool to improve grammar, clarity, and presentation. All scientific ideas, methods, experiments, analyses, and conclusions are our own.

Reproducibility Statement

The method is described in Section 3.1 and Section 3, with full details of the graph-conditioned generator in Section B, the expansion modes in Section C and Algorithm 1, and field optimization in Section A.2. All source models and datasets used in our experiments are publicly available.

Unless stated otherwise, results are averaged over three seeds, and single-run results are identified in Section 4. The full implementation will be open-sourced upon acceptance.

Broader Impact

By generating effective initializations directly from architectural metadata, this work can reduce the computational and environmental cost of training large language models, broadening access for resource-constrained researchers and institutions. While lowering the barrier to producing capable models may amplify existing misuse risks, these concerns are shared across efficiency-oriented research and are best addressed through complementary efforts in alignment and responsible deployment. Our method does not introduce new model capabilities; it offers a more efficient path to the same endpoint.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:52004855.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Zhao Changsheng, Chang Ernie, Liu Zechun, Chang Chia-Jung, Wen Wei, Lai Chen, Cao Sheng, Tian Yuandong, Krishnamoorthi Raghuraman, Shi Yangyang, and Chandra Vikas. Mobilellm-r1: Exploring the limits of sub-billion language model reasoners with open training recipes. *arXiv preprint arXiv:2509.24945*, 2025.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2BERT: Towards reusable pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2134–2148, May 2022.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer, 2016. URL https://arxiv.org/abs/1511.05641.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient LLM pre-training. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=FXJDcriMYH.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of BERT by progressively stacking. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2337–2346. PMLR, 09–15 Jun 2019.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon

Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

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

Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Xuying Meng, Siqi Fan, Peng Han, Jing Li, Li Du, Bowen Qin, Zheng Zhang, Aixin Sun, and Yequan Wang. Flm-101b: An open llm and how to train it with $100k budget, 2025.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1144.

William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A. Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints, 2022. URL https://arxiv.org/abs/2209.12892.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Mohammad Samragh, Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. Scaling smart: Accelerating large language model pre-training with small model initialization. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 1–13. PMLR, 14 Dec 2024.

Konstantin Schürholt, Boris Knyazev, Xavier Giró i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018. URL https://arxiv.org/abs/1803.09820.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion, 2024a.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *The Eleventh International Conference on Learning Representations*, 2023.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. LEMON: Lossless model expansion. In *The Twelfth International Conference on Learning Representations*, 2024b.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024.

A Implementation Details

A.1 Weight Extraction

The extraction script in the supplementary material traverses `named_modules`, combines each weight matrix with its bias as described in Section 3.1, and emits coordinate–weight pairs stored in bf16. Normalization parameters and embedding matrices are handled separately (Section C.5).

A.2 Field Optimization

We train f_θ with AdamW (Loshchilov & Hutter, 2017) using a OneCycle learning-rate schedule, gradient clipping, and mixed-precision training, reshuffling the coordinate–value dataset every epoch. The log-cosh objective is implemented in the numerically stable form

$$\log\cosh(x) = x + \mathrm{softplus}(-2x) - \log 2, \tag{7}$$

which avoids overflow for large residuals.

B Cross-Architecture Graph Conditioning

Architecture graph. Each architecture is represented as a directed graph $\mathcal{G} = (V, E)$ whose nodes are functional components: the embedding, normalization layers, attention projections (`q`, `k`, `v`, `o`), MLP projections (`gate`, `up`, `down`), the final normalization layer, and the output head. Edges follow the forward-pass connectivity through successive transformer blocks.

Node features. Each node v has a feature vector $\mathbf{x}_v \in \mathbb{R}^{D_{\text{node}}}$ with three groups of features:

1. **Categorical:** one-hot encodings of functional type and architecture family;
2. **Structural:** hidden dimension, layer count, attention-head and key-value-head counts, head dimension, intermediate size, MLP expansion ratio, and indicators for gated activations, fused QKV projections, and bias terms;
3. **Geometric:** log-normalized matrix height and width, their ratios to the hidden dimension, and a fractional chunk index for partitioned embeddings.

Continuous structural features are normalized to $[0, 1]$.

Table 5: **Expansion modes used in the ablation.** W_s: source weights; $f_\theta@(R, C)$: field decoded at shape $R \times C$; $[:R_s, :C_s]$: source-sized top-left block of the target. Rescale: fan-in variance rescaling (Appendix C.1). Spectral: spectral rank restoration (Appendix C.2).

Mode	Construction of W_t
`inr`	$f_\theta@(R_t, C_t)$
`inr_rescale`	$f_\theta@(R_t, C_t)$ + rescale + spectral
`inr_merge`	$(1-\beta)\,\mathrm{tile}(W_s) + \beta\, f_\theta@(R_t, C_t)$
`inr_tile_merge`	$(1-\beta)\,\mathrm{tile}(f_\theta@(R_s, C_s)) + \beta\, f_\theta@(R_t, C_t)$
`inr_blend`	$(1-\beta)\,\mathrm{tile}(W_s) + \beta\, f_\theta@(R_t, C_t)$ + rescale + spectral
`inr_extend`	W_s in $[:R_s, :C_s]$; $f_\theta@(R_t, C_t)$ elsewhere
`inr_src`	$f_\theta@(R_s, C_s)$ in $[:R_s, :C_s]$; $f_\theta@(R_t, C_t)$ elsewhere

Graph encoder. A two-layer GELU MLP projects the node features, followed by $K = 3$ rounds of mean-aggregation message passing (Equation (5)), where each $\phi^{(k)}$ is a two-layer MLP mapping $2H_g \to H_g$ with $H_g = 128$. The global representation $\mathbf{z}_{\mathrm{arch}} \in \mathbb{R}^{64}$ is obtained by mean-pooling the final node states, followed by LayerNorm and a linear projection; a separate LayerNorm–linear head produces the per-node representation $\mathbf{z}_v \in \mathbb{R}^{64}$.

FiLM conditioning. The FiLM parameters $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$ in Equation (6) are predicted by one linear layer per residual block, with weights initialized to zero and biases to $[\mathbf{1}; \mathbf{0}]$, giving identity scaling and zero shift at initialization.

Type decomposition. Each parameter tensor is assigned to one of ten functional types: `attn_q`, `attn_k`, `attn_v`, `attn_out`, `mlp_gate`, `mlp_up`, `mlp_down`, `embedding`, `lm_head`, and `normalization`. Embedding and output-head matrices are partitioned into row chunks of size 4096, with a fractional chunk index included in the conditioning, and normalization vectors are reshaped to $\mathbb{R}^{H\times 1}$. Fused QKV tensors are split along the output dimension according to the head allocation before assignment to the individual attention types.

Training across architectures. For each functional type, matrices from all source architectures are combined into a shared coordinate–target pool and globally shuffled each epoch, so that mini-batches contain multiple architectures and depths. The graph encoder is evaluated once per unique architecture in a mini-batch, and the resulting $(\mathbf{z}_{\mathrm{arch}}, \mathbf{z}_v)$ embeddings are reused for the corresponding coordinates. The coordinate-grid parameters of the graph-conditioned INR use a $5\times$ higher learning rate than the MLP parameters, with OneCycleLR and cosine annealing (Smith, 2018).

C Expansion Modes

Table 5 summarizes the expansion variants used in the experiments. All modes first decode the field on the target grid, $W = s \cdot \sinh(f_\theta(\mathrm{grid}_{R_t \times C_t}))$, and differ in whether the result is used directly, statistically corrected, or combined with source weights.

C.1 Fan-In Variance Rescaling

For $y = Wx$ with $W \in \mathbb{R}^{O\times I}$ and i.i.d. inputs of variance σ_x^2, each output has variance $I\,\sigma_w^2\,\sigma_x^2$. Preserving activation variance when the fan-in changes from C_s to C_t requires

$$C_t\,\sigma_{w,t}^2 = C_s\,\sigma_{w,s}^2 \quad\Longrightarrow\quad \sigma_{w,t} = \sigma_{w,s}\sqrt{\frac{C_s}{C_t}}. \tag{8}$$

We apply this correction around the mean of the decoded tensor so that the mean is unchanged:

$$W_{\mathrm{t}} \leftarrow \mu + (W_{\mathrm{t}} - \mu)\sqrt{\frac{C_s}{C_t}}, \qquad \mu = \mathrm{mean}(W_{\mathrm{t}}). \tag{9}$$

C.2 Spectral Rank Restoration

Because the frequency content of f_θ is bounded by its Fourier features, evaluating the field on a denser grid can produce correlated entries and near-degenerate singular directions. We decompose

$W_{\mathrm{t}} = U\Sigma V^\top$ and define

$$k_s = \left|\{i : \sigma_i > \epsilon\sigma_1\}\right|, \qquad \epsilon = 0.01, \qquad k_t = \lfloor \alpha \min(R_t, C_t) \rfloor, \quad \alpha \in [0.5, 0.9]. \tag{10}$$

The Frobenius-norm deficit is

$$\Delta = \max\left(0,\ \frac{R_t C_t}{C_t} - \sum_{i=1}^{k_s} \sigma_i^2\right). \tag{11}$$

For directions $i \in \{k_s + 1, \ldots, k_t\}$, we set

$$\lambda_i = \sigma_{k_s}\sqrt{1 - 0.95\,\frac{i - k_s}{k_t - k_s}}, \qquad \eta = \sqrt{\frac{\Delta}{\sum_{i=k_s+1}^{k_t} \lambda_i^2}}, \tag{12}$$

and correct the tensor as

$$W_{\mathrm{t}} \leftarrow W_{\mathrm{t}} + \eta \sum_{i=k_s+1}^{k_t} \lambda_i\, \mathbf{u}_i \mathbf{v}_i^\top. \tag{13}$$

The correction acts on directions outside the dominant singular subspace of the decoded field.

C.3 Source Blending

The `inr_blend` mode combines the tiled source tensor with the field decoded at the target resolution:

$$W_{\mathrm{t}} = (1 - \beta)\,\mathrm{tile}(W_{\mathrm{s}}, R_t, C_t) + \beta\, f_\theta(\mathrm{grid}_{R_t \times C_t}), \tag{14}$$

where $\mathrm{tile}(W_{\mathrm{s}}, R_t, C_t)$ repeats the source tensor along both axes and trims it to the target shape, extending standard tiling to non-integer target dimensions. We use $\beta \in [0.01, 0.1]$ for the MobileLLM expansion and $\beta = 0.5$ for the BERT tile-plus-noise control. Fan-in rescaling and spectral restoration are then applied as in `inr_rescale`; the ablation-only `inr_merge` variant uses the same blend without these corrections.

C.4 Source Preservation

The `inr_extend` mode keeps the source tensor in the source-sized region and uses the field for the new dimensions:

$$W_{\mathrm{t}} = \begin{pmatrix} W_{\mathrm{s}} & F_{\mathrm{cols}} \\ F_{\mathrm{rows}} & F_{\mathrm{corner}} \end{pmatrix}, \tag{15}$$

where $W_{\mathrm{s}} \in \mathbb{R}^{R_s \times C_s}$ is copied exactly from the source checkpoint and the remaining blocks come from the field decoded on the full target grid. The ablation-only `inr_src` variant instead places the field reconstruction at source resolution in the source-sized block. For an attention projection $W_Q \in \mathbb{R}^{n_h d_h \times d}$, F_{cols} connects new input channels to existing output channels, F_{rows} introduces new output channels from existing inputs, and F_{corner} connects new inputs to new outputs; expanded MLP projections follow the same interpretation.

C.5 Normalization and Embedding Expansion

Normalization parameters. All normalization scale vectors are stacked into $\mathbf{N} \in \mathbb{R}^{N_{\mathrm{norms}} \times d_s}$ and modeled with a shared field, which is queried at $[N_{\mathrm{norms}}, d_t]$ to generate parameters at the target hidden dimension. Under `inr_extend`, the original channels $[0, d_s)$ are restored from the source checkpoint.

Embeddings. The embedding matrix $E \in \mathbb{R}^{V \times d_s}$ is divided into contiguous vocabulary chunks modeled with chunk-conditioned fields. Each field is queried at the target hidden dimension d_t, and `lm_head` is tied to the expanded embedding matrix.

C.6 Expansion Algorithm

Algorithm 1 summarizes the expansion modes used in the main experiments; the additional ablation variants are defined in Table 5.

Algorithm 1 WINR expansion

Require: Source state dict, target configuration, fitted fields, expansion mode
for each 2D parameter tensor with target shape (R_t, C_t) **do**
  $W \leftarrow s \cdot \sinh\big(f_\theta(\mathrm{grid}_{R_t \times C_t})\big)$
  if mode $=$ `inr_rescale` **then**
    $W \leftarrow$ `fan_in_rescale`(W, C_s, C_t)
    $W \leftarrow$ `spectral_restore`(W, C_t)
  else if mode $=$ `inr_blend` **then**
    $W \leftarrow (1-\beta)\,\mathrm{tile}(W_s, R_t, C_t) + \beta W$
    Apply fan-in rescaling and spectral restoration
  else if mode $=$ `inr_extend` **then**
    $W[: R_s, : C_s] \leftarrow W_s$
  end if
end for
Expand normalization parameters with the shared normalization field
Expand embeddings with the chunk-conditioned fields
Tie `lm_head` to the expanded embedding

Table 6: Per-tensor dimensional changes for MobileLLM-R1-140M-base → MobileLLM-R1-360M.

Tensor	Source	Target	$\sqrt{C_s/C_t}$
`q_proj`, `o_proj`	576×576	1024×1024	0.750
`k_proj`, `v_proj`	192×576	512×1024	0.750
`gate_proj`, `up_proj`	2048×576	2816×1024	0.750
`down_proj`	576×2048	1024×2816	0.853
norms	576	1024	—
embedding	$V \times 576$	$V \times 1024$	—

C.7 MobileLLM-R1 140M → 360M Dimensions

MobileLLM-R1-140M-base and MobileLLM-R1-360M both have $L = 15$ transformer layers, so this experiment isolates width expansion. Table 6 lists the tensor dimensions and fan-in correction factors.

D Additional Experimental Results

D.1 Gemma-3-270M Training Dynamics

Figure 9 reports per-step trajectories for a single cross-architecture Gemma-3-270M run. At 15K steps, WINR reaches perplexities of 194.9 on WikiText, 571.3 on Text8, and 334.2 on LAMBADA, compared with 198.0, 572.3, and 344.3 for random initialization. On ARC-Challenge, random initialization decreases from 24.7% at 3K steps to 21.2% at 12K steps, whereas WINR recovers after a dip at 9K steps and reaches 24.9% at 15K steps.

D.2 Expansion-Strategy Ablation

Table 7 reports the full numerical comparison of expansion modes, and the accompanying curves show WikiText-2 perplexity as a function of downstream training budget for the strategies analyzed in the main text.

D.3 Field-Capacity Ablation

Field capacity is expressed relative to the number of represented parameters, and fitting time is measured from the start of optimization. Larger fields reach high-fidelity reconstruction substantially faster: the $1\times$ field reaches cosine 0.98 after 180.3 s but not 0.99, whereas the $2\times$ and $4\times$ fields reach 0.99 after 187.5 s and 149.7 s. The $0.5\times$ field reaches neither threshold within the evaluated optimization budget.

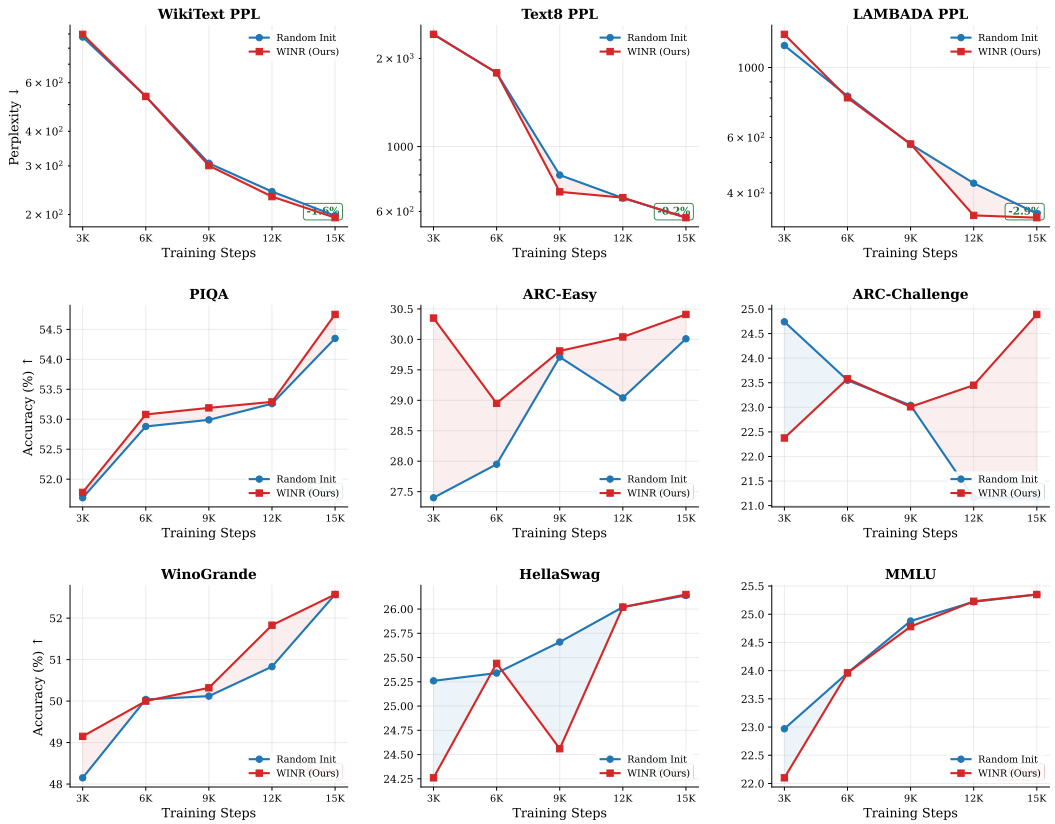


Figure 9: **Cross-architecture pre-training on Gemma-3-270M** initialized from a zoo-conditioned WINR field vs. random initialization, both trained on SlimPajama for 15K steps (single run). Top: perplexity (log scale, ↓); middle and bottom: zero-shot accuracy (↑). The largest gains at the final checkpoint are on LAMBADA PPL (-2.9%) and ARC-Challenge ($+3.7$ pp).

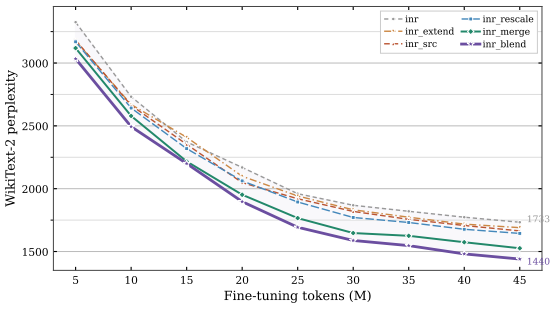


Figure 10: **Expansion strategy ablation for MobileLLM-R1-140M-base → MobileLLM-R1-360M.** WikiText-2 perplexity vs. fine-tuning token budget for each INR expansion strategy. `inr_blend` achieves the lowest perplexity at every budget.

Table 7: **Expansion strategy ablation.** WikiText-2 PPL (↓) at three budgets. Δ: relative reduction vs. `inr` at 45 M.

Strategy	5M	25M	45M	Δ (%)
`inr`	3325	1960	1733	—
`inr_extend`	3177	1942	1690	2.5
`inr_src`	3182	1920	1664	4.0
`inr_rescale`	3169	1894	1644	5.1
`inr_merge`	3119	1766	1526	11.9
`inr_tile_merge`	3107	1766	1527	11.9
`inr_blend`	**3034**	**1693**	**1440**	**16.9**

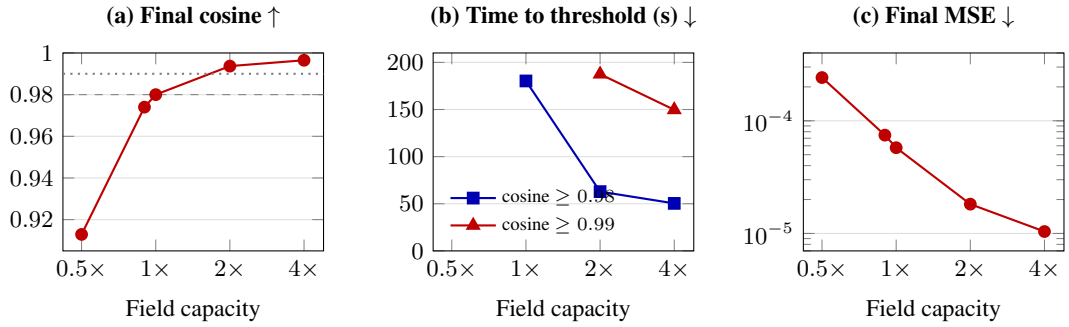


Figure 11: **Field capacity vs. reconstruction and fitting time.** Capacity is relative to the size of the represented weights. (a, c) Final reconstruction cosine and MSE; dashed and dotted lines in (a) mark cosine 0.98 and 0.99; the $0.5\times$ field is reported at epoch 2000 (553.6 s), all others at epoch 500. (b) Wall-clock time, measured from epoch 0, at which cosine first reaches 0.98 or 0.99; missing points indicate the threshold was not reached ($0.5\times$ and $0.9\times$ reach neither; $1\times$ does not reach 0.99).