CROSSGMN: GRAPH METANETWORKS FOR CROSS-ARCHITECTURE WEIGHT-SPACE TRANSFORMATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight-space networks operate directly on parameters of other neural networks, enabling tasks such as predicting model properties, editing trained models, and generating weights. Equivariance has emerged as a key design principle for these networks, owing to weight-space symmetries such as neuron permutations. However, existing equivariant weight-space architectures have primarily been studied for transformations that preserve the network architecture. In contrast, many practical transformations, including model compression and upscaling, map a trained source network into a target network with a different architecture. In this setting, the source and target permutation symmetries act on different parameter spaces, making equivariance less straightforward to formulate. Our key idea for addressing this mismatch is to reformulate cross-architecture operators with two inputs: a trained source network and an initialization of the target network. This lets us define equivariant cross-architecture operators that refine the initialization of the target network using information from the source network, while being invariant to source-network permutations and equivariant to target-network permutations. Based on this formulation, we introduce CrossGMN, a graph metanetwork that jointly processes both networks through symmetry-preserving cross-network message passing. We prove CrossGMN is universal for continuous cross-architecture operators on compact sets under a general-position assumption. We further evaluate CrossGMN for model compression, where it predicts the parameters of a smaller network to accelerate subsequent knowledge distillation. Across 2-D and 3-D INRs and image classification with MLPs, CNNs, and Vision Transformers, CrossGMN speeds up distillation by up to 8.89×, transfers across datasets without retraining (3.78×), and a single model can accelerate compression from heterogeneous source architectures into a common target architecture.

1 INTRODUCTION

Weight-space learning studies neural architectures that operate directly on the parameters of other neural networks, enabling tasks such as predicting model properties, editing trained models, and generating weights (Unterthiner et al., 2020; Eilertsen et al., 2020; Schürholt et al., 2021; 2022a; 2024). A central property of weight spaces is their permutation symmetries, resulting in many different model parameterizations that represent the same function. This has motivated the development of permutation-equivariant weight-space architectures (Navon et al., 2023; Zhou et al., 2023a;b; 2024; Lim et al., 2024; Kofinas et al., 2024; Tran et al., 2024; 2025; Vo et al., 2025), which represent the current state of the art for learning over weights in many tasks.

Many practical transformations of neural network weights do not preserve the network architecture. Model compression maps a trained network into a smaller one for deployment under memory, latency, or compute constraints, through structured pruning (Li et al., 2017) or knowledge distillation (Hinton et al., 2015; Gou et al., 2021). Cross-architecture knowledge transfer moves the behavior of a trained network into a structurally different one (Ahn et al., 2019; Liu et al., 2024; Czyzewski, 2021). Model upscaling initializes a wider or deeper network from a trained smaller one, allowing training to continue from the transferred parameters rather than from scratch (Chen et al., 2016; Wei et al., 2016; Chen et al., 2022; Wang et al., 2023; Pham et al., 2024). All of these map the weights of one architecture into the parameter space of another, and most of them are realized today by a separate optimization process run from scratch for every source network. Learning

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.

such a transformation instead would replace this per-network optimization with a single feedforward pass: one weight-space model is trained once, and every new source network is then transformed in a fraction of a second. Existing equivariant weight-space architectures, however, have exclusively addressed mappings whose input and output lie in the *same* weight space. The cross-architecture setting is fundamentally different: the source and target networks carry distinct symmetry groups acting on distinct parameter spaces, so it is not even clear what equivariance should mean. Designing equivariant cross-architecture weight-space models is thus a central yet overlooked challenge in weight-space learning.

Our key observation is that cross-architecture transformations typically come with a natural second input. Whether the target is smaller or larger than the source, the procedure always begins from an *initialized* target network, which is then optimized against the trained source network, as in knowledge distillation and its variants (Hinton et al., 2015; Gou et al., 2021; Nasser et al., 2024). The transformation is therefore a function of two arguments: the trained source network and an initialization of the target. Importantly, taking both as input resolves the difficulty above. The initialization specifies the target architecture and carries the target symmetry group, so the operator has a well-defined action on each of its arguments: it should be invariant to symmetries of the source network and equivariant to symmetries of the target. This leads us to study equivariant *cross-architecture operators* $F : \mathcal{V}_{\mathrm{src}} \times \mathcal{V}_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$.

With the right class of operators in place, the remaining question is how to parameterize it. We introduce *CrossGMN*, a graph metanetwork for learning such operators. CrossGMN represents each of the two networks as a parameter graph, joins them with cross edges, and performs message passing over the resulting joint graph. Information can thus flow between the source and the target while the two permutation symmetries are respected independently. We prove that CrossGMN is a universal approximator of continuous equivariant cross-architecture operators on compact sets under a general-position assumption, and extend this result to finite collections of source architectures mapping into a common target weight space.

We evaluate CrossGMN for model compression through knowledge distillation. Given a trained teacher (source network) and an initialization for a smaller student (target network), CrossGMN predicts refined student parameters intended to reduce the number of distillation steps needed to match the teacher. We evaluate width, depth, and combined compressions across 2-D and 3-D INRs (Sitzmann et al., 2020) and MNIST (LeCun et al., 1998)/CIFAR-10 (Krizhevsky, 2009) classifiers, on architectures including MLPs, CNNs, and Vision Transformers (Dosovitskiy et al., 2021). Across these settings, CrossGMN accelerates subsequent distillation for initializing compressed students, achieving speedups of up to $8.89\times$ on MNIST INRs and $6.85\times$ on ModelNet (Wu et al., 2015) INRs. CrossGMN also generalizes out of distribution: a model trained on MNIST INRs transfers to FashionMNIST (Xiao et al., 2017), achieving up to $3.78\times$ speedup without retraining. Moreover, a single CrossGMN can compress heterogeneous CIFAR-10 MLP teacher architectures into a common smaller student architecture and generalize to unseen teacher architectures.

In summary, our contributions are: (1) formulating cross-architecture weight-space learning as learning equivariant operators on a source network and a target initialization, which makes the problem's symmetry structure well-defined; (2) introducing CrossGMN, a graph metanetwork for learning equivariant transformations between different neural-network weight spaces; (3) establishing its universality for continuous cross-architecture operators, including settings with multiple source architectures; and (4) demonstrating its effectiveness for model compression, where it accelerates knowledge distillation across diverse architectures and out-of-distribution settings.

2 PRELIMINARIES

We introduce the notation and background required to define cross-architecture weight-space operators. For the remainder of the manuscript, we focus our theoretical analysis on fully connected MLPs, while evaluating our method empirically on a broader range of architectures, including CNNs and Transformers. The corresponding formal treatment for these architectures is provided in Appendix B.2.3–B.2.4.

Weight spaces. An MLP architecture $A = (\boldsymbol{d}, \sigma)$ with L layers is specified by layer widths $\boldsymbol{d} = (d_0, \ldots, d_L)$ and an activation function σ. Its parameters are

$$\boldsymbol{v} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_L), \qquad \boldsymbol{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}, \quad \boldsymbol{b}_\ell \in \mathbb{R}^{d_\ell}.$$

The corresponding realized function $f_{\boldsymbol{v}} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ is

$$f_{\boldsymbol{v}}(\boldsymbol{x}) = \boldsymbol{W}_L \sigma(\cdots \sigma(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) \cdots) + \boldsymbol{b}_L.$$

Definition 1 (Weight space)**.** For an architecture $A = (\boldsymbol{d}, \sigma)$, its weight space is $\mathcal{V}_A := \bigoplus_{\ell=1}^{L} \left(\mathbb{R}^{d_\ell \times d_{\ell-1}} \oplus \mathbb{R}^{d_\ell}\right)$, where, $\oplus$ denotes a direct sum.

Throughout the paper, we consider a source architecture A_{src} and a target architecture A_{tgt}, with corresponding weight spaces $\mathcal{V}_{\mathrm{src}}$ and $\mathcal{V}_{\mathrm{tgt}}$. Their hidden widths and depths may differ, but we assume the same input and output dimensions.

Permutation symmetries. Permuting hidden neurons changes the parameterization of an MLP while preserving its realized function. We formalize these transformations through the following group action.

Definition 2 (Permutation action on weight space)**.** For an architecture $A = (\boldsymbol{d}, \sigma)$, let $G_A := S_{d_1} \times \cdots \times S_{d_{L-1}}$ be its hidden-neuron permutation group. For $g = (\tau_1, \ldots, \tau_{L-1}) \in G_A$, set $\tau_0, \tau_L = e$ and let $\boldsymbol{P}_{\tau_\ell}$ denote the corresponding permutation matrix. The action $\rho_A(g) : \mathcal{V}_A \to \mathcal{V}_A$ is defined by $\boldsymbol{W}'_\ell = \boldsymbol{P}_{\tau_\ell}^\top \boldsymbol{W}_\ell \boldsymbol{P}_{\tau_{\ell-1}}$ and $\boldsymbol{b}'_\ell = \boldsymbol{P}_{\tau_\ell}^\top \boldsymbol{b}_\ell$ for $1 \le \ell \le L$. By construction, $f_{\rho_A(g)\boldsymbol{v}} = f_{\boldsymbol{v}}$ for every $g \in G_A$.

Definition 3 (Permutation-invariant and equivariant weight-space networks)**.** Let A be an architecture, and let G_A act on its weight space $\mathcal{V}_A$ through ρ_A as in Definition 2. A weight-space network is a parameterized map $\Psi_\theta : \mathcal{V}_A \to Y$, where G_A acts on Y through a representation ρ_Y. The network is G_A-*equivariant* if $\Psi_\theta(\rho_A(g)\boldsymbol{v}) = \rho_Y(g)\Psi_\theta(\boldsymbol{v})$ for all $g \in G_A$ and $\boldsymbol{v} \in \mathcal{V}_A$. It is G_A-*invariant* if $\Psi_\theta(\rho_A(g)\boldsymbol{v}) = \Psi_\theta(\boldsymbol{v})$ for all $g \in G_A$ and $\boldsymbol{v} \in \mathcal{V}_A$.

In the next section, we consider the cross-architecture setting in which the source and target networks belong to different weight spaces with independent permutation symmetries.

3 Cross-Architecture Weight-Space Transformations

Let A_{src} and A_{tgt} be two architectures with weight spaces $\mathcal{V}_{\mathrm{src}}$ and $\mathcal{V}_{\mathrm{tgt}}$, and permutation groups G_{src} and G_{tgt}.

Limitations of predicting the target directly from the source. One could directly map source parameters to target parameters, $T : \mathcal{V}_{\mathrm{src}} \longrightarrow \mathcal{V}_{\mathrm{tgt}}$. However, this formulation has two difficulties. First, its equivariance structure is unclear: because the source and target have different permutation groups acting on different weight spaces, T must be invariant to source permutations while selecting a representative among equivalent target parameterizations. Second, natural source-to-target maps may be discontinuous and thus cannot be uniformly approximated arbitrarily well by continuous neural networks. Appendix C.1 gives a simple setting of model compression where mapping each larger network to an optimal smaller one is discontinuous.

Instead, we formulate the problem as refinement from an initialization of the target network; this gives rise to more natural continuous cross-architecture operators. This leads to the following definition.

Definition 4 (Cross-architecture weight-space operator)**.** A cross-architecture weight-space operator is a map

$$F : \mathcal{V}_{\mathrm{src}} \times \mathcal{V}_{\mathrm{tgt}} \longrightarrow \mathcal{V}_{\mathrm{tgt}}, \qquad (\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) \longmapsto F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}). \tag{1}$$

The second input $\boldsymbol{v}_{\mathrm{init}}$ both specifies a concrete parameterization in the target weight space and provides the parameters that the operator can refine. We say that F is equivariant with respect to the source and target permutations if

$$F(g_{\mathrm{src}} \cdot \boldsymbol{v}_{\mathrm{src}}, g_{\mathrm{tgt}} \cdot \boldsymbol{v}_{\mathrm{init}}) = g_{\mathrm{tgt}} \cdot F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) \quad \forall (g_{\mathrm{src}}, g_{\mathrm{tgt}}) \in G_{\mathrm{src}} \times G_{\mathrm{tgt}} \tag{2}$$

A natural and practically important instance of the operator class above is knowledge distillation (Hinton et al., 2015). In its simplest setting, a target network, called the *student*, is trained to match the outputs of a source network, called the *teacher*. In our setting, we denote their architectures by $A_T = A_{\mathrm{src}}$ and $A_S = A_{\mathrm{tgt}}$, their weight spaces by $\mathcal{V}_T$ and $\mathcal{V}_S$, the trained teacher by $\boldsymbol{v}_T$, and the student by $\boldsymbol{v}_S$. Distillation then minimizes

$$L(\boldsymbol{v}_T, \boldsymbol{v}_S) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\ell(f_{\boldsymbol{v}_T}(\boldsymbol{x}), f_{\boldsymbol{v}_S}(\boldsymbol{x}))\right], \tag{3}$$

where ℓ measures the discrepancy between teacher and student outputs.

Let $\Phi_k(\boldsymbol{v}_T, \boldsymbol{v}_0)$ denote the student parameters obtained after k distillation steps initialized at $\boldsymbol{v}_0$. For SGD (Bottou, 2012) and Adam (Kingma & Ba, 2015) with fixed hyperparameters and minibatches, Φ_k is a continuous equivariant cross-architecture operator in the sense of Definition 4 for every fixed k, provided the student loss gradient is jointly continuous. Appendix C.2 gives the formal statement and proof.

4 CrossGMN

To learn the cross-architecture operators defined in the previous section (Definition 4), we introduce CrossGMN. Section 4.1 presents its architecture, and Section 4.2 establishes its universality for continuous cross-architecture operators.

4.1 Architecture

For parameters $\boldsymbol{v}$, let $\mathcal{P}(\boldsymbol{v}) = (V, E)$ denote the parameter graph constructed as in GMN (Lim et al., 2024). For an MLP, neurons form the nodes, weights are represented as directed edges between consecutive layers, and biases as node features. The full graph construction and feature specification are given in Appendix B.2.2.

Given source and target MLP architectures, let $\boldsymbol{v}_{\mathrm{src}}$ denote the source-network parameters and $\boldsymbol{v}_{\mathrm{init}}$ an initialization of the target network. Let $\mathcal{P}(\boldsymbol{v}_{\mathrm{src}}) = (V^{\mathrm{src}}, E^{\mathrm{src}})$ and $\mathcal{P}(\boldsymbol{v}_{\mathrm{init}}) = (V^{\mathrm{tgt}}, E^{\mathrm{tgt}})$ denote their parameter graphs. We define the *joint graph* as

$$\mathcal{J}(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) := \left(V^{\mathrm{src}} \sqcup V^{\mathrm{tgt}},\ E^{\mathrm{src}} \sqcup E^{\mathrm{tgt}} \sqcup E_{\times}\right), \tag{4}$$

where $E_{\times}$ denotes the set of cross edges. For the joint graph to respect both symmetries, $E_{\times}$ must be invariant under the joint action of $G_{\mathrm{src}} \times G_{\mathrm{tgt}}$ on neurons: whenever $(u, v) \in E_{\times}$, also $(g \cdot u,\ h \cdot v) \in E_{\times}$ for all $g \in G_{\mathrm{src}}$ and $h \in G_{\mathrm{tgt}}$. We construct $E_{\times}$ in two steps. We first determine how the neurons of a source layer are joined to those of a target layer, which is dictated by this invariance requirement. We then choose which pairs of layers to join, which is a free design choice.

Joining two layers. Consider a source hidden layer and a target hidden layer. Since the neurons of each can be permuted independently, the only nonempty set of edges between them that is invariant to both permutations is the complete bipartite one, connecting every source neuron to every target neuron. We call this a *layerwise join*, following the graph join of two disjoint graphs (Cardoso et al., 2013). Input and output neurons, in contrast, are indexed by the coordinates of the realized function's input and output, and are not permuted. Since the source and target share input and output dimensions, we join them with the sparser *positionwise join*: the i-th source input neuron is connected only to the i-th target input neuron, and likewise for output neurons.

Choosing which layers to join. Because each layerwise join is invariant on its own, any choice of which source and target hidden layers to join yields a symmetry-preserving $E_{\times}$. We call this choice a *cross-edge policy*. The most permissive policy, *all-to-all*, joins every source hidden layer to every target hidden layer, giving nm layerwise joins for networks with n and m hidden layers. To avoid this quadratic growth with depth, we use the following *strided* policy.

Definition 5 (Strided cross-edge policy)**.** For two networks with $n \geq m$ hidden layers, the *strided* policy partitions the n hidden layers of the deeper network into m consecutive blocks of near-equal size, and joins the k-th hidden layer of the shallower network to every layer in the k-th block. Each hidden layer of the deeper network is thus joined to exactly one hidden layer of the shallower network, for a total of n layerwise joins. For example, with $n = 10$ and $m = 5$, the k-th hidden layer of the shallower network is joined to layers $2k - 1$ and $2k$ of the deeper network, giving 10 joins instead of the 50 required by all-to-all. The formal construction is given in Appendix B.2.1.

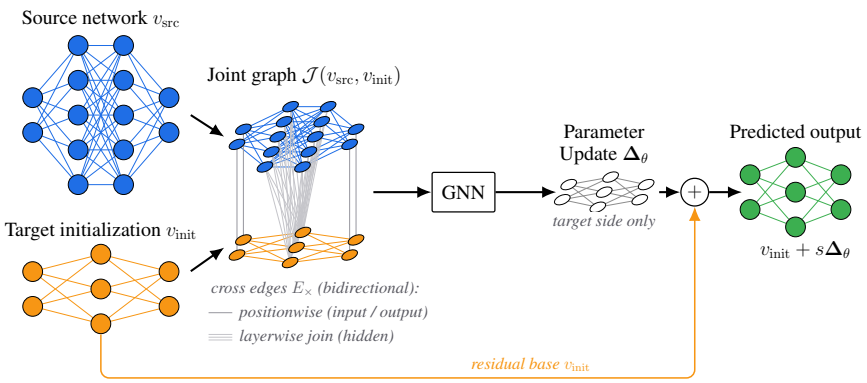


Figure 1: **CrossGMN.** The source parameters v_{src} and an initialization v_{init} of the target network are represented as parameter graphs and combined into the joint graph $\mathcal{J}(v_{\text{src}}, v_{\text{init}})$. A GNN operates on the joint graph, and its outputs corresponding to the target network form the parameter update $\boldsymbol{\Delta}_\theta$, yielding the CrossGMN output defined in Definition 6.

Importantly, the strided policy retains the benefits of all-to-all connectivity while using fewer cross edges. In Section 4.2, we establish universality for the strided construction, and our experiments later show that it performs comparably to all-to-all connectivity in practice (Table 6). We now have all the components needed to define CrossGMN.

Definition 6 (CrossGMN)**.** Given the joint graph $\mathcal{J}(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}})$, let GMN_θ denote a graph metanetwork with learnable parameters θ operating on the joint graph, and let Π_{tgt} denote the map that extracts the target-side parameters and arranges them into a vector in the target weight space. We define the resulting parameter update as

$$\boldsymbol{\Delta}_\theta(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}) = \Pi_{\text{tgt}}(\text{GMN}_\theta(\mathcal{J}(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}))) \,.$$

The *CrossGMN* operator is then

$$\text{CrossGMN}_\theta(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}) = \boldsymbol{v}_{\text{init}} + s\, \boldsymbol{\Delta}_\theta(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}), \tag{5}$$

where $s > 0$ is a fixed residual-scale hyperparameter. Figure 1 illustrates CrossGMN.

By construction, CrossGMN is invariant to source permutations and equivariant to target permutations, as required by Definition 4. Message passing on the joint graph preserves both symmetry actions, while Π_{tgt} discards the source and retains the target, yielding the desired invariance and equivariance. A formal statement and proof are given in Proposition 2 in Appendix B.2.5.

4.2 Expressivity

Symmetry-preserving architectures restrict the hypothesis space and may therefore limit approximation power, making expressivity analysis particularly important in this setting. We establish that CrossGMN is universal for arbitrary continuous cross-architecture operators satisfying the symmetry in equation 2. Our result applies whenever the cross-edge connectivity contains all edges of the strided policy in Definition 5.

Theorem 1 (Universality of CrossGMN)**.** *Let A_{src} and A_{tgt} be fixed MLP architectures, and let $K_{\text{src}} \subset \mathcal{V}_{A_{\text{src}}}$ and $K_{\text{tgt}} \subset \mathcal{V}_{A_{\text{tgt}}}$ be compact sets of weights in general position.*[1] *Let $F : K_{\text{src}} \times K_{\text{tgt}} \to \mathcal{V}_{\text{tgt}}$ be a continuous cross-architecture operator in the sense of Definition 4. Then, for every $\varepsilon > 0$, there exists a CrossGMN$_\theta$ such that*

$$\sup_{(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}) \in K_{\text{src}} \times K_{\text{tgt}}} \|\textit{CrossGMN}_\theta(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}}) - F(\boldsymbol{v}_{\text{src}}, \boldsymbol{v}_{\text{init}})\| < \varepsilon.$$

In particular, this theorem provides an approximation guarantee for CrossGMN applied to the knowledge distillation operator Φ_k introduced in Section 3.

[1] We use the same general-position assumption as Dayan et al. (2026); for an MLP, this means that within each hidden layer, no two neurons have the same bias value.

Table 1: **INR compression.** We report median speedup + IQR (in brackets) relative to the best pruning-based anchor initialization. The last two columns report the percentage of held-out teachers for which CrossGMN accelerates distillation, and the percentage for which CrossGMN fails to reach the target within the budget.

Dataset	Reduction axis	Architecture change	Speedup	$> 1\times$ (%)	Failed (%)
MNIST	Width	$[2, 32, 32, 1] \to [2, 16, 16, 1]$	$2.20\times_{[1.2,\,3.9]}$	83	2
	Width	$[2, 32, 32, 1] \to [2, 8, 8, 1]$	$5.20\times_{[1.1,\,12.5]}$	76	18
	Width	$[2, 32, 32, 1] \to [2, 4, 4, 1]$	$4.10\times_{[0.0,\,14.1]}$	62	35
	Depth	$[2, 32, 32, 1] \to [2, 32, 1]$	$4.07\times_{[2.2,\,6.6]}$	90	7
	Width+depth	$[2, 32, 32, 1] \to [2, 16, 1]$	$8.89\times_{[3.6,\,16.8]}$	86	13
ModelNet40	Width	$[3, 48, 48, 48, 48, 1] \to [3, 24, 24, 24, 24, 1]$	$1.11\times_{[0.9,\,1.7]}$	57	18
	Depth	$[3, 48, 48, 48, 48, 1] \to [3, 48, 48, 1]$	$6.85\times_{[1.0,\,19.8]}$	74	12
	Depth	$[3, 48, 48, 48, 48, 1] \to [3, 48, 1]$	$2.88\times_{[2.0,\,4.0]}$	96	0
	Width+depth	$[3, 48, 48, 48, 48, 1] \to [3, 24, 24, 1]$	$2.46\times_{[0.4,\,4.8]}$	68	23

Proof idea. The proof reduces the cross-architecture setting to the fixed-architecture universality results of Dayan et al. (2026), who showed that prominent permutation-equivariant weight-space architectures, including DWSNets (Navon et al., 2023) and GMNs (Lim et al., 2024), have equivalent expressive power and are universal on compact general-position sets for invariant functionals and equivariant operators. Specifically, fixing $\boldsymbol{v}_{\mathrm{init}}$ makes $\boldsymbol{v}_{\mathrm{src}} \mapsto F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$ source-invariant, while fixing $\boldsymbol{v}_{\mathrm{src}}$ makes $\boldsymbol{v}_{\mathrm{init}} \mapsto F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$ target-equivariant. We approximate these components using the corresponding invariant and equivariant universality results and combine them through the cross edges. The full proof is given in Appendix C.3.

A single CrossGMN for heterogeneous architectures. The universality result extends to any finite collection of source architectures $A_{\mathrm{src}}^{(1)}, \ldots, A_{\mathrm{src}}^{(m)}$ sharing a common target architecture A_{tgt}, provided the source architecture is identifiable from the joint graph. The formal statement and proof are given in Appendix C.4, and we evaluate this setting empirically in Section 5.2.

5 Experiments

We evaluate CrossGMN for model compression through knowledge distillation, a practical instance of the cross-architecture operators discussed in Section 3. Following that example, we refer to the source network as the teacher and the target network as the student. We measure whether student initializations predicted by CrossGMN accelerate subsequent distillation toward their teachers. Our experiments cover several model families and compression settings, including transfer across datasets and to unseen source architectures. Full experimental details are provided in Appendix D.1–D.7.

5.1 Experimental setup and evaluation

Training and evaluation protocol. For each compression setting, we train CrossGMN on a population of trained teachers and student initializations using the distillation objective in equation 3. Given a teacher $\boldsymbol{v}_T$, CrossGMN predicts $\hat{\boldsymbol{v}_S} = \mathrm{CrossGMN}_\theta(\boldsymbol{v}_T, \boldsymbol{v}_0)$, where $\boldsymbol{v}_0$ is an initialization of the target network as defined in Section 3, which we refer to as the *anchor* throughout the compression experiments.

We initialize the anchor $\boldsymbol{v}_0$ by applying structured pruning methods to the teacher $\boldsymbol{v}_T$, as this provides a stronger starting point for knowledge distillation than random initialization with little additional compression overhead (Xu et al., 2024). For each experiment, we select the structured-pruning method that yields the lowest loss at initialization among the evaluated methods, providing a strong anchor for CrossGMN to improve upon. We compare CrossGMN with the selected anchor itself and with other structured-pruning baselines, including magnitude-based (Xu et al., 2024), activation-based (Hu et al., 2016), and OBS/OBC pruning (Hassibi & Stork, 1992; Frantar & Alistarh, 2022). Appendix D.2 details the anchor and baseline initialization methods.

At evaluation, we measure the distillation speedup of CrossGMN relative to the anchor separately for each held-out teacher. We first distill the anchor $\boldsymbol{v}_0$ until its relevant validation metric plateaus,[2]

[2]We define a plateau as no improvement greater than $\epsilon = 10^{-4}$ for $P = 50$ consecutive evaluations, following the standard early-stopping principle (Prechelt, 1998).

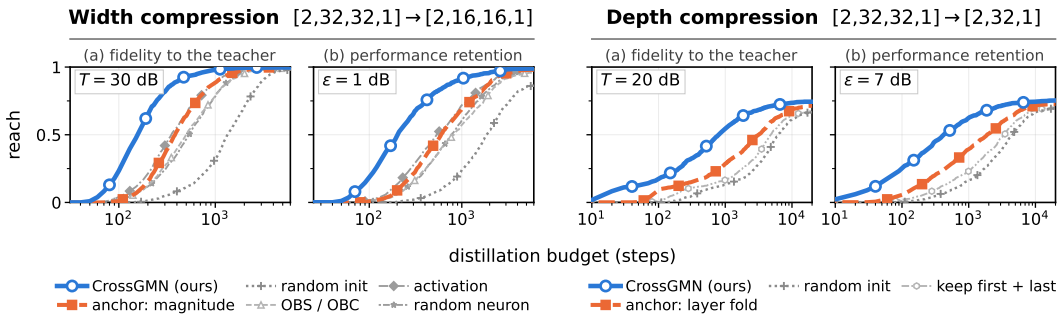


Figure 2: **Reach-budget plots for MNIST INRs.** Fraction of 1,000 held-out teachers whose students reach a prescribed performance threshold defined relative to the teacher as a function of distillation budget (log scale), comparing CrossGMN with its anchor and other baselines (Appendix D.3). Each compression setting includes two panels: **(a)** *Fidelity to the teacher*, measured by $\mathrm{PSNR}(\text{student}, \text{teacher}) \geq T$ dB, and **(b)** *Student performance retention*, measured by $\mathrm{PSNR}(\text{student}, \text{image}) \geq \mathrm{PSNR}(\text{teacher}, \text{image}) - \varepsilon$. The thresholds T and ε are chosen separately for each compression setting to account for differences in student architecture capacity. See Section 5.1 for the full description of the reach-budget plots.

and denote the attained value by $q^{\star}_{\boldsymbol{v}_T}$. We then distill the CrossGMN prediction $\hat{\boldsymbol{v}}_S$ toward the same teacher under a fixed budget. The *per-teacher speedup* $S_{\boldsymbol{v}_T}$ is the ratio between the numbers of distillation steps required by $\boldsymbol{v}_0$ and $\hat{\boldsymbol{v}}_S$ to reach $q^{\star}_{\boldsymbol{v}_T}$. Thus, $S_{\boldsymbol{v}_T} > 1$ indicates that CrossGMN accelerates distillation relative to the anchor. If $\hat{\boldsymbol{v}}_S$ does not reach $q^{\star}_{\boldsymbol{v}_T}$ within the budget, we count the teacher as a failure and set $S_{\boldsymbol{v}_T} = 0$. We report the median speedup over all held-out teachers together with two complementary statistics: the acceleration rate, defined as the fraction of teachers with $S_{\boldsymbol{v}_T} > 1$, and the failure rate, defined as the fraction of teachers with $S_{\boldsymbol{v}_T} = 0$. We additionally report wall-clock speedup, including the cost of the CrossGMN forward pass.

Reach–budget plots. A speedup metric relative to an anchor may be less informative when the anchor itself is weak. We therefore complement speedup metrics with *reach–budget plots*, which evaluate whether the student reaches prescribed performance defined relative to the teacher. Each plot shows the fraction of teachers whose students reach a fixed target as the distillation budget increases. We report two measures: *fidelity reach*, which tracks how quickly students match their teachers' functions, and *performance-retention reach*, which tracks how quickly they attain a prescribed level of their teachers' task performance. Figure 2 shows both measures for representative settings; Appendices D.2 and D.9 provide the full protocol and plots for all settings.

Teacher model families. We evaluate compression on trained-model populations [3], including 2-D SIREN implicit neural representations (INRs) (Sitzmann et al., 2020) fitted to MNIST images (LeCun et al., 1998), 3-D SIREN INRs fitted to ModelNet40 shapes (Wu et al., 2015), MLP and Vision Transformer (ViT) classifiers (Dosovitskiy et al., 2021) on CIFAR-10 (Krizhevsky, 2009), and CNN classifiers on MNIST from the model zoo of Schürholt et al. (2022b).

5.2 CrossGMN distillation speedup across model families and datasets

INR datasets. For MNIST, we use the dataset of Navon et al. (2023), containing one SIREN INR per image with teacher architecture $[2, 32, 32, 1]$ (1,185 parameters). For ModelNet40, we use the SIREN dataset of 3-D shape distance fields from Shamsian et al. (2024), with teacher architecture $[3, 48, 48, 48, 48, 1]$ (7,297 parameters). CrossGMN is trained and evaluated on disjoint teacher subsets for each dataset; Appendix D.3–D.4 provides full details. Table 1 reports distillation speedups across compression settings, and Figure 2 shows reach–budget curves for representative MNIST width and depth reductions, with wall-clock speedups reported below.

Wall-clock speedup. We additionally report wall-clock speedup, including graph construction and the CrossGMN forward pass. For the MNIST INR settings in Table 1, CrossGMN inference takes

[3] To justify amortization, each model population should span diverse functions; classifier diversity is quantified in Appendix D.11.

Table 2: **Classifier compression.** Median distillation speedup + IQR (in brackets) relative to the best pruning-based anchor initialization; columns as defined in Table 1.

Family	Reduction axis	Architecture change	Speedup	$> 1\times$ (%)	Failed (%)
MLP	Width	$[3072, 64, 64, 64, 64, 10] \to [3072, 32, 32, 32, 32, 10]$	$1.04\times_{[0.8,\,1.4]}$	54	5
	Depth	$[3072, 64, 64, 64, 64, 10] \to [3072, 64, 64, 10]$	$1.25\times_{[0.7,\,2.3]}$	60	3
	Width+depth	$[3072, 64, 64, 64, 64, 10] \to [3072, 32, 32, 10]$	$1.18\times_{[0.8,\,1.8]}$	59	3
ViT	Depth	blocks $4 \to 2$	$1.21\times_{[1.1,\,1.3]}$	95	0
	Attention dimension	$48 \to 24$	$0.74\times_{[0.6,\,0.9]}$	7	5
	Head count	heads $2 \to 1$	$0.86\times_{[0.8,\,1.0]}$	16	0
	Head count+depth	heads $2 \to 1$, blocks $4 \to 2$	$1.18\times_{[1.1,\,1.3]}$	92	0
CNN	Width	conv $[8, 6, 4] \to [4, 3, 2]$	$1.17\times_{[0.9,\,1.4]}$	68	1

18–20 ms per teacher on an NVIDIA A100, compared with 1.8–2.4 ms per distillation step. This overhead is small relative to the overall distillation budget: width compression from $[2, 32, 32, 1]$ to $[2, 8, 8, 1]$, with a budget of $2{,}480$ steps, achieves a $4.99\times$ wall-clock speedup versus $5.20\times$ in distillation steps, while joint width–depth compression to $[2, 16, 1]$, with a budget of $2{,}750$ steps, achieves $8.36\times$ versus $8.89\times$. Full timing details and results are provided in Appendices D.2 and D.10.

Classifier datasets. We conduct separate compression experiments on MLP, ViT, and CNN model zoos. For MLPs and ViTs, we construct separate datasets of models trained on CIFAR-10, with $5{,}000$ training and $1{,}000$ test models per dataset. The MLPs have architecture $[3072, 64, 64, 64, 64, 10]$ (209,802 parameters), while the ViTs have four Transformer blocks, two attention heads, hidden dimension 48, and FFN width 192 (115,930 parameters). For CNNs, we use the existing model zoo of Schürholt et al. (2022b), containing $2{,}560$ MNIST classifiers with three convolutional layers of widths $[8, 6, 4]$ ($2{,}464$ parameters). Table 2 reports the compression settings and results; Appendix D.5–D.7 provides dataset construction and training details.

Heterogeneous-architecture compression. When a model collection spans multiple architectures, training a separate compression operator for each reduces the benefits of amortization. To evaluate this setting, we train a single CrossGMN on four CIFAR-10 MLP teacher architectures, all mapped to a common student architecture. Table 3 reports results for held-out teachers from each architecture and for the pooled population.

Discussion. As seen in Table 1, CrossGMN consistently accelerates distillation across INRs, achieving significant speedups. For classifiers, Table 2 shows more modest gains that depend on the compression setting: CrossGMN improves distillation for MLPs, CNNs and the ViT depth reductions, while reducing the ViT attention dimension or head count does not yield a speedup. The results in Table 3 show that CrossGMN achieves positive speedups across heterogeneous architectures. Taken together, the results provide evidence that model compression can be learned and amortized directly in weight space, and suggest that CrossGMN is a suitable architecture for studying such learned cross-architecture transformations.

Table 3: **Heterogeneous-architecture compression.** A single CrossGMN maps all four source architectures to $[3072, 8, 10]$.

Teacher	Speedup	$> 1\times$ (%)	Failed (%)
$[3072, 16, 10]$	$1.28\times_{[0.7,\,1.9]}$	60	12
$[3072, 16, 16, 10]$	$1.07\times_{[0.7,\,1.6]}$	55	14
$[3072, 32, 10]$	$1.40\times_{[0.8,\,3.4]}$	65	6
$[3072, 32, 32, 10]$	$1.54\times_{[0.7,\,2.9]}$	66	10
Pooled	$1.28\times_{[0.7,\,2.3]}$	62	10

5.3 Out-of-Distribution Generalization

A desirable property of weight-space models is to become increasingly agnostic to the underlying network architecture, rather than requiring retraining for each specific model family. We therefore evaluate CrossGMN under two out-of-distribution shifts: cross-dataset transfer and unseen source architectures. Table 4 reports cross-dataset transfer, while Table 5 evaluates unseen architectures. These results support CrossGMN's ability to transfer across datasets and provide evidence of partial generalization to unseen architectures.

Table 4: **Transfer across datasets.** The CrossGMN models trained on the MNIST INR settings of Table 1 are applied unchanged to FashionMNIST INRs of the same architectures.

Architecture change	Speedup	$> 1\times$ (%)	Failed (%)
$[2, 32, 32, 1] \to [2, 8, 8, 1]$	$2.18\times_{[0.0,\ 5.8]}$	61	33
$[2, 32, 32, 1] \to [2, 32, 1]$	$3.09\times_{[1.5,\ 6.1]}$	86	6
$[2, 32, 32, 1] \to [2, 16, 1]$	$3.78\times_{[1.1,\ 10.3]}$	76	17

Table 5: **Generalization to unseen architectures.** The CrossGMN checkpoint used in Section 5.2 is applied without retraining to unseen wider and deeper teacher architectures.

Source architecture	Speedup	$> 1\times$ (%)	Failed (%)
$[3072, 64, 10]$	$1.38\times_{[0.5,\ 3.2]}$	61	11
$[3072, 16, 16, 16, 16, 10]$	$0.87\times_{[0.4,\ 1.3]}$	41	17
Pooled	$1.06\times_{[0.5,\ 1.8]}$	51	14

5.4 Ablations: the role of cross edges

We ablate the main architectural and training choices of CrossGMN, including teacher–student communication, cross-edge policy, student initialization, supervision objective, teacher-population size, weight-space backbone, and operator capacity. Table 6 focuses on the communication ablation, which highlights two main findings: removing cross edges or replacing them with FiLM conditioning on a pooled teacher summary (Perez et al., 2018) reduces distillation speedup, supporting direct equivariant neuron-level communication; and the strided policy of Definition 5 performs comparably to all-to-all connectivity while using fewer cross edges. The remaining ablations are reported in Appendix D.8.

Table 6: **Teacher–student communication.** CrossGMN ablations on MNIST INR width reduction $[2, 32, 32, 1] \to [2, 8, 8, 1]$, same compression setting as in Table 1.

Variant	Speedup	$> 1\times$ (%)	Failed (%)
Strided (Definition 5)	$5.20\times_{[1.1,\ 12.5]}$	76	18
All-to-all (Section 4.1)	$5.13\times_{[0.9,\ 11.6]}$	75	18
No cross edges	$3.18\times_{[0.4,\ 8.6]}$	67	24
FiLM conditioning	$3.22\times_{[0.4,\ 9.1]}$	68	24

6 Related Work

Weight-space learning treats neural-network parameters as structured data (Han et al., 2026). The pioneering works that started exploring this direction are Eilertsen et al. (2020); Unterthiner et al. (2020); Schürholt et al. (2021). Later approaches use symmetry considerations and include equivariant weight-space networks (Navon et al., 2023; Zhou et al., 2023a;b; Tran et al., 2025) and graph-based models (Lim et al., 2024; Kofinas et al., 2024), with extensions to additional symmetries (Kalogeropoulos et al., 2024; Tran et al., 2024). Importantly, these symmetry-aware architectures operate within a single weight space, whereas we study transformations between distinct source and target weight spaces. Related parameter-generation methods either condition on a target architecture (Zhang et al., 2019; Knyazev et al., 2021; Liu et al., 2019) or learn distributions over trained-model populations (Schürholt et al., 2022a; 2024; Erkoç et al., 2023; Wang et al., 2024; Soro et al., 2025). They generate new weights rather than transforming a particular trained network into another weight space. An extended discussion is provided in Appendix A.

7 Conclusion

We explored learning transformations between the weight spaces of different neural architectures and introduced CrossGMN, an equivariant framework for learning such cross-architecture operators. CrossGMN is a graph metanetwork that operates jointly over the source and target weight spaces, using a trained source network to refine an initialization of the target network. We proved universality for continuous cross-architecture operators and evaluated the framework on model compression with knowledge distillation. Across INRs and classifiers, CrossGMN reduces subsequent distillation steps for both width and depth compression, while supporting reuse across source architectures, transfer across datasets, and partial generalization to unseen source architectures. These results show that equivariant models can learn architecture-changing transformations directly in weight space.

Limitations and future work. Our current setting requires access to source parameters, assumes compatible input and output spaces, and is evaluated only at moderate scale (up to roughly 200k parameters). Scaling to larger models may require sparse or blockwise joint graphs and more efficient cross-network communication. Future work could also extend the operator view beyond compression to settings such as model upscaling and cross-family transfer.

AI USE STATEMENT

In this work, we used generative AI tools to write code for generating synthetic datasets (model populations), assist in revising the presentation of proofs, provide feedback on research methodology and experiments, and implement methods. We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, formulate mathematical claims, provide critical ingredients for proving those claims, perform translation, clean or reformat datasets, conduct qualitative or thematic data analysis, or interpret results. These latter tasks were performed by the authors.

Additionally, we used generative AI tools to create and modify scientific figures, create and edit software code, prepare research artifacts including LaTeX source and tables, summarize and analyze existing literature, identify research topics and gaps, and edit the manuscript to improve readability.

We reviewed and verified all AI-assisted work. Code, mathematical exposition, and manuscript drafts underwent iterative author inspection and correction. We take responsibility for the final content of this work, including all text, claims, code, figures, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

Our experiments use publicly available benchmark datasets and model zoos, alongside models that we train on these benchmarks. This work involves no human-subject studies or collection of personal or sensitive data. CrossGMN is a methodological contribution to learning cross-architecture weight-space transformations, evaluated on model compression. We do not identify specific ethical concerns arising from the experiments reported here; broader applications should be assessed in their intended deployment contexts.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our theoretical and empirical results in the supplementary material. Complete proofs of the theoretical results are given in Appendices B.2.5 and C. Detailed descriptions of the datasets, model architectures, training procedures, hyperparameters, baselines, and evaluation protocols are provided in Appendix D.1–D.7. We will release the code in the camera-ready version.

REFERENCES

Sungsoo Ahn, Shell Xu Hu, Andreas C. Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9155–9163, 2019.

Guy Bar-Shalom, Fabrizio Frasca, Yaniv Galron, Yftah Ziser, and Haggai Maron. Beyond token probes: Hallucination detection via activation tensors with act-vit. In *Advances in Neural Information Processing Systems*, volume 38, pp. 95218–95248, 2025. doi: 10.52202/085713-2865.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: tricks of the trade: second edition*, pp. 421–436. Springer, 2012.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL `https://doi.org/10.1137/16M1080173`.

Domingos M Cardoso, Maria Aguieiras A de Freitas, Enide Andrade Martins, and María Robbiano. Spectra of graphs obtained by a generalization of the join graph operation. *Discrete Mathematics*, 313(5):733–741, 2013.

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, 2022.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

Maciej A Czyzewski. Transfer learning between different architectures via weights injection. *arXiv preprint arXiv:2101.02757*, 2021.

Adir Dayan, Yam Eitan, and Haggai Maron. On the expressive power of permutation-equivariant weight-space networks. In *International Conference on Machine Learning (ICML)*, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. In *ECAI 2020: 24th European Conference on Artificial Intelligence, 29 August–8 September 2020, Santiago de Compostela, Spain–Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, pp. 1119–1126. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2020.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14254–14264. IEEE, 2023.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, 2022.

Yoav Gelberg, Yam Eitan, Aviv Navon, Aviv Shamsian, Theo Putterman, Michael M. Bronstein, and Haggai Maron. Gradmetanet: An equivariant architecture for learning on gradients. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5155.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819, 2021.

Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, Rose Yu, Haggai Maron, Yanfang Ye, Lu Yin, et al. A survey of weight space learning: Understanding, representation, and generation. *arXiv preprint arXiv:2603.10090*, 2026.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pp. 164–171, 1992.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. doi: 10.1016/0893-6080(91)90009-T. URL https://doi.org/10.1016/0893-6080(91)90009-T.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 106800–106840. Curran Associates, Inc., 2024. doi: 10.52202/079017-3391. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c13d5a10028586fdc15ee7da97b7563f-Paper-Conference.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Boris Knyazev, Michal Drozdzal, Graham W. Taylor, and Adriana Romero-Soriano. Parameter prediction for unseen deep architectures. In *Advances in Neural Information Processing Systems*, 2021.

Miltiadis Miltos Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees G Snoek, and David Zhang. Graph neural networks for learning equivariant representations of neural networks. In *International Conference on Learning Representations*, volume 2024, pp. 45363–45381, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, 2017.

Derek Lim, Haggai Maron, Marc T Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 11430–11458, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2feafd84b23a52b3fa434bc6d7682256-Paper-Conference.pdf.

Yewei Liu, Xiyuan Wang, and Muhan Zhang. Meta pruning via graph metanetworks: A meta learning framework for network pruning. *arXiv preprint arXiv:2506.12041*, 2025. URL https://arxiv.org/abs/2506.12041.

Yufan Liu, Jiajiong Cao, Bing Li, Weiming Hu, Jingting Ding, Liang Li, and Stephen Maybank. Cross-architecture knowledge distillation. *International Journal of Computer Vision*, 132(8): 2798–2824, 2024. doi: 10.1007/s11263-024-02002-0. URL https://doi.org/10.1007/s11263-024-02002-0.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *2019 IEEE/CVF international conference on computer vision (ICCV)*, pp. 3295–3304. IEEE, 2019.

Sahar Almahfouz Nasser, Nihar Gupte, and Amit Sethi. Reverse knowledge distillation: Training a large model using a small one for retinal image matching on limited data. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 7778–7787, 2024. doi: 10.1109/WACV57701.2024.00760.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pp. 25790–25816. PMLR, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Chau Pham, Piotr Teterwak, Soren Nelson, and Bryan A. Plummer. Mixturegrowth: Growing neural networks by recombining learned parameters. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2788–2797, 2024.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5. URL https://doi.org/10.1016/0041-5553(64)90137-5.

Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural networks*, 11(4):761–767, 1998.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27906–27920, 2022a.

Konstantin Schürholt, Diyar Taskiran, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Model zoos: A dataset of diverse populations of neural network models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022b.

Konstantin Schürholt, Michael W. Mahoney, and Damian Borth. Towards scalable and versatile weight space learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43947–43966, 2024.

Aviv Shamsian, Aviv Navon, David W. Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved generalization of weight space networks via augmentations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 44378–44393, 2024.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

Bedionita Soro, Bruno Andreis, Hayeon Lee, Wonyong Jeong, Song Chong, Frank Hutter, and Sung Ju Hwang. Diffusion-based neural network weights generation. In *International Conference on Learning Representations*, volume 2025, pp. 99644–99675, 2025.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Neural Networks for Machine Learning, Coursera, 2012. URL `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. Lecture notes.

Viet-Hoang Tran, Thieu N. Vo, Tho Tran-Huu, An T. Nguyen, and Tan M. Nguyen. Monomial matrix group equivariant neural functional networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Viet-Hoang Tran, Thieu N. Vo, An Nguyen, Tho-Huu Tran, Minh-Khoi Nguyen-Nhat, Thanh Tran, Duy-Tung Pham, and Tan Nguyen. Equivariant neural functional networks for transformers. In *International Conference on Learning Representations*, 2025.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

Thieu Vo, Hoang V. Tran, Tho Tran Huu, An Nguyen The, Thanh Tran, Minh-Khoi Nguyen-Nhat, Duy-Tung Pham, and Tan Minh Nguyen. Equivariant polynomial functional networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61689–61744, 2025.

Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

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

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International conference on machine learning*, pp. 564–572. PMLR, 2016.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Lingjie Liu, Zhuang Liu, et al. Initializing models with larger ones. In *International Conference on Learning Representations*, volume 2024, pp. 14336–14351, 2024.

Zhuoqian Yang, Mathieu Salzmann, and Sabine Süsstrunk. Weight space representation learning via neural field adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17734–17743, 2026.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. In *Advances in Neural Information Processing Systems*, volume 36, 2023a.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Neural functional transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023b.

Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

APPENDIX CONTENTS

A Extended Related Work **16**

B Notation and Architecture-Specific Symmetries **16**

B.1 Notation 16

B.2 Parameter graphs and induced permutation actions 17

B.2.1 Joint graphs and cross-edge policies 17

B.2.2 Multilayer perceptrons 18

B.2.3 Convolutional networks 19

B.2.4 Transformers 20

B.2.5 Equivariance of the joint graph 22

C Theoretical Results **23**

C.1 Discontinuity of mapping a larger network to an optimal smaller one 23

C.2 Finite-step distillation is a cross-architecture operator 25

C.3 Universal approximation of cross-architecture operators 27

C.4 One CrossGMN for finitely many source architectures 30

C.5 Scope for CNN and Transformer weight spaces 31

D Experimental Details **31**

D.1 CrossGMN implementation 31

D.2 Evaluation protocol 32

D.3 MNIST and FashionMNIST INR compression 35

D.4 ModelNet40 3-D INR compression 35

D.5 CIFAR-10 MLP compression 36

D.6 CIFAR-10 Vision Transformer compression 36

D.7 MNIST CNN compression 37

D.8 Ablations 37

D.9 Reach–budget plots 37

D.10 Complete results 42

D.11 Classifier teacher-population diversity 43

A Extended Related Work

Weight-space learning. Weight-space learning treats neural-network parameters as structured data for learning, representation, and manipulation (Han et al., 2026). DWSNets (Navon et al., 2023) and Neural Functional Networks (Zhou et al., 2023a) construct permutation-equivariant layers directly in weight space, while Neural Functional Transformers (Zhou et al., 2023b) use attention. Graph Metanetworks (Lim et al., 2024) and Neural Graph models (Kofinas et al., 2024) instead represent networks as graphs and apply message passing. Related work also considers additional symmetries (Kalogeropoulos et al., 2024; Tran et al., 2024), structured parameterizations such as LoRA weights (Yang et al., 2026), and related neural artifacts such as gradients (Gelberg et al., 2025) and activations (Bar-Shalom et al., 2025). Universal Neural Functionals extend equivariant weight-space models to general architectures (Zhou et al., 2024), while related approaches extend neural functionals to Transformers and polynomial weight-space features (Tran et al., 2025; Vo et al., 2025). These models have also been used to learn operators on trained networks, including optimization (Kofinas et al., 2024) and pruning-oriented editing (Liu et al., 2025). Existing expressivity results compare such architectures over a single weight space (Dayan et al., 2026). In contrast, CrossGMN studies learned operators between distinct source and target weight spaces with independent permutation actions.

Parameter generation. Parameter-generation methods amortize weight generation across architectures, tasks, or model populations. Graph HyperNetworks (Zhang et al., 2019; Knyazev et al., 2021) and MetaPruning (Liu et al., 2019) take a target architecture as input and predict its weights. Hyper-representations (Schürholt et al., 2022a; 2024) and diffusion-based generators (Erkoç et al., 2023; Wang et al., 2024; Soro et al., 2025) instead learn a distribution over a population of trained models and sample new weights, optionally conditioned on the task or dataset. These methods generate weights for an architecture or population rather than transforming a particular trained source network from one weight space into another.

Source-conditioned architecture changes. Net2Net provides analytic function-preserving transformations that widen or deepen a network (Chen et al., 2016), and model-growth methods extend this approach to pretrained Transformers (Chen et al., 2022). LiGO (Wang et al., 2023) learns a structured linear map from a smaller pretrained model to an initialization of a larger one. While these methods use analytic growth rules or optimize a growth map for a given pretrained source model, CrossGMN learns a shared operator across a population of source networks and reuses it on unseen sources without fitting a new operator for each model.

Compression and distillation. Structured pruning constructs a smaller network by removing entire units or channels and reusing the retained weights (Li et al., 2017). Knowledge distillation trains a student to match a teacher's outputs (Hinton et al., 2015; Gou et al., 2021). FitNets additionally uses intermediate teacher representations as hints to train thinner, deeper students (Romero et al., 2015). Distillation can also start from weights selected from the teacher (Xu et al., 2024). CrossGMN fits between these stages: it jointly processes the teacher and the structured-pruning initialization (the anchor) to predict a refined student initialization, from which distillation proceeds toward the same teacher. The refinement is learned across a population of teachers to accelerate this subsequent optimization.

B Notation and Architecture-Specific Symmetries

This section records the notation shared by the theoretical and experimental appendices and makes precise which permutation actions are respected by the parameter graphs used for MLPs, CNNs, and Transformers. The main universality theorem is stated for MLPs; the final paragraph of Appendix C explains what transfers to the other families.

B.1 Notation

The theoretical sections use the terms *source* and *target*, whereas the compression with knowledge distillation experiments use *teacher* and *student*. Table 7 fixes the correspondence. We write $g \cdot \boldsymbol{v}$ for

the action $\rho_A(g)\boldsymbol{v}$ of Definition 2. In the experiments, the *anchor* is simply the target initialization supplied to the operator.

Table 7: Correspondence between the theoretical and experimental notation.

Object	Theory	Compression experiments
Architectures	$A_{\mathrm{src}}, A_{\mathrm{tgt}}$	teacher A_T, student A_S
Weight spaces	$\mathcal{V}_{\mathrm{src}}, \mathcal{V}_{\mathrm{tgt}}$	$\mathcal{V}_T, \mathcal{V}_S$
Trained input	$\boldsymbol{v}_{\mathrm{src}}$	teacher $\boldsymbol{v}_T$
Target initialization	$\boldsymbol{v}_{\mathrm{init}}$	anchor $\boldsymbol{v}_0$
Operator output	$F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$	predicted student $\hat{\boldsymbol{v}}_S$
Symmetry groups	$G_{\mathrm{src}}, G_{\mathrm{tgt}}$	G_T, G_S
Fixed-step optimizer	$\Phi_k(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$	$\Phi_k(\boldsymbol{v}_T, \boldsymbol{v}_0)$

B.2 Parameter graphs and induced permutation actions

Parameter graph. CrossGMN uses the parameter-graph construction of Lim et al. (2024). We state only the structure needed for the symmetry arguments and for reproducing the implementation.

Definition 7 (Parameter graph)**.** For parameters $\boldsymbol{v} \in \mathcal{V}_A$, the parameter graph $\mathcal{P}_A(\boldsymbol{v}) = (V, E)$ is a typed directed multigraph. Unit nodes represent neurons, channels, or attention coordinates. Auxiliary nodes represent parameter vectors not indexed by two units, such as biases and the gain and shift parameters of normalization layers. Every learnable scalar is stored on either a node or an edge: a matrix entry $[W]_{ij}$ is carried by the edge $j \to i$; the entries of a convolutional kernel give parallel edges between its input and output channels; and bias or normalization entries are stored on their unit node or on an edge from the corresponding auxiliary node. A fixed *parameter-to-graph encoding* specifies this assignment and the correspondence used to decode graph outputs into network parameters (Appendix D.1). Nodes carry layer and source/target labels; classifier nodes also carry node type, canonical input/output indices, and optional architecture-family and activation-type labels. Edges carry their scalar value, layer index, and role (source-internal, target-internal, source-to-target, or target-to-source in the joint graph), with an optional architecture-family label for classifiers.

Following Lim et al. (2024, Section 2.2 and Appendix C.2), a *neural DAG automorphism* preserves edge incidence, fixes input, output, and bias nodes, and preserves weight-sharing constraints. We use the term *neural graph automorphism* for the corresponding relabelings of our parameter and joint graphs, which may include reciprocal edges. We require these relabelings to fix auxiliary nodes and preserve all structural node and edge features, as in Lim et al. (2024, Appendix C.6). Parameter values are permuted with their nodes or edges, rather than held numerically fixed. Message passing with shared functions is equivariant to these relabelings. The layerwise joins of Section 4.1 are preserved under independent relabelings of their two endpoint layers, while positionwise joins connect only canonically indexed boundary nodes. Hence the joint graph preserves the source and target actions independently.

For an MLP, the automorphisms of the standard parameter graph are precisely the hidden-neuron permutations in Definition 2 (Lim et al., 2024, Proposition 5). We first specify the shared cross-edge policy, then give the full definitions for MLPs, CNNs, and Transformers.

B.2.1 Joint graphs and cross-edge policies

Joint graph and layerwise joins. For source and target parameter graphs $\mathcal{P}_{\mathrm{src}}(\boldsymbol{v}_{\mathrm{src}}) = (V^{\mathrm{src}}, E^{\mathrm{src}})$ and $\mathcal{P}_{\mathrm{tgt}}(\boldsymbol{v}_{\mathrm{init}}) = (V^{\mathrm{tgt}}, E^{\mathrm{tgt}})$, the joint graph is

$$\mathcal{J}(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) = \big(V^{\mathrm{src}} \sqcup V^{\mathrm{tgt}}, E^{\mathrm{src}} \sqcup E^{\mathrm{tgt}} \sqcup E_{\times}\big), \tag{6}$$

as in Equation 4. For node sets U, W on opposite sides, write

$$\mathrm{Join}(U, W) = (U \times W) \cup (W \times U)$$

for the bidirectional layerwise join of Section 4.1, with each ordered pair representing a directed cross edge. Cross-edge structural features are constant within each directed join. For CNNs and Transformers, cross edges have scalar value 0 and their direction role; auxiliary nodes retain only their internal connections, as in Appendix D.1.

Positionwise boundary joins. For canonically indexed node sets $U = \{u_i : i \in [q]\}$ and $W = \{w_i : i \in [q]\}$, where $[q] = \{1, \dots, q\}$, define

$$\mathrm{Pos}(U, W) = \{(u_i, w_i), (w_i, u_i) : i \in [q]\}.$$

Source and target share input and output dimensions. Following Section 4.1, their boundary cross edges are

$$E_{\times,\partial} = \mathrm{Pos}(U^{\mathrm{src}}_{\mathrm{in}}, U^{\mathrm{tgt}}_{\mathrm{in}}) \cup \mathrm{Pos}(U^{\mathrm{src}}_{\mathrm{out}}, U^{\mathrm{tgt}}_{\mathrm{out}}). \tag{7}$$

These nodes are fixed by the permutation actions, so positionwise joins preserve the required symmetries. Internal layer and block boundaries remain hidden-unit sets and use layerwise joins.

Strided cross-edge policy. We give the indexed construction of Definition 5. Let $n, m \geq 0$ be the numbers of source and target hidden layers, indexed from 0. For $n \geq m \geq 1$, define

$$\begin{aligned} a_i &= \mathrm{round}\left(\frac{in}{m}\right), \qquad b_i = \max\left\{\mathrm{round}\left(\frac{(i+1)n}{m}\right), a_i + 1\right\}, \\ \mathcal{M}_{n,m} &= \bigcup_{i=0}^{m-1} \big(\{a_i, \dots, b_i - 1\} \times \{i\}\big), \end{aligned} \tag{8}$$

where round denotes nearest-integer rounding with a fixed tie convention. Each target layer i is paired with the source layers in the nonempty block $\{a_i, \dots, b_i - 1\}$; these blocks partition the source hidden layers. For $n < m$, set $\mathcal{M}_{n,m} = \{(i, j) : (j, i) \in \mathcal{M}_{m,n}\}$, so pairs always list the source index first. If either network has no hidden layers, set $\mathcal{M}_{n,m} = \varnothing$. In particular, $\mathcal{M}_{n,n} = \{(i, i) : 0 \leq i < n\}$ gives the shared-index policy for width-only compression. For Transformers, the same matching rule is applied to blocks, with joins defined below by submodule type.

B.2.2 Multilayer perceptrons

Weight space. Let $A = (\boldsymbol{d}, \sigma)$ have widths $\boldsymbol{d} = (d_0, \dots, d_L)$ and entrywise activation σ. As in Definition 1, its parameters $\boldsymbol{v} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \dots, \boldsymbol{W}_L, \boldsymbol{b}_L)$ lie in

$$\mathcal{V}_A = \bigoplus_{\ell=1}^{L} \left(\mathbb{R}^{d_\ell \times d_{\ell-1}} \oplus \mathbb{R}^{d_\ell}\right).$$

Realized function. For $\boldsymbol{x} \in \mathbb{R}^{d_0}$, set

$$\boldsymbol{h}_0 = \boldsymbol{x}, \qquad \boldsymbol{h}_\ell = \sigma(\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1} + \boldsymbol{b}_\ell) \quad (1 \leq \ell < L), \qquad f_{\boldsymbol{v}}(\boldsymbol{x}) = \boldsymbol{W}_L \boldsymbol{h}_{L-1} + \boldsymbol{b}_L.$$

Thus hidden layers apply σ and the output layer is affine, as in Section 2.

Permutation symmetries. The group $G_A = \prod_{\ell=1}^{L-1} S_{d_\ell}$ acts as in Definition 2. For $g = (\tau_1, \dots, \tau_{L-1})$, set $\boldsymbol{P}_{\tau_0} = I_{d_0}$ and $\boldsymbol{P}_{\tau_L} = I_{d_L}$, and define

$$\boldsymbol{W}'_\ell = \boldsymbol{P}_{\tau_\ell}^\top \boldsymbol{W}_\ell \boldsymbol{P}_{\tau_{\ell-1}}, \qquad \boldsymbol{b}'_\ell = \boldsymbol{P}_{\tau_\ell}^\top \boldsymbol{b}_\ell, \qquad 1 \leq \ell \leq L.$$

Since σ commutes with permutations, the transformed hidden states satisfy $\boldsymbol{h}'_\ell = \boldsymbol{P}_{\tau_\ell}^\top \boldsymbol{h}_\ell$, giving $f_{\rho_A(g)\boldsymbol{v}} = f_{\boldsymbol{v}}$. The same permutations relabel the hidden nodes and their incident edges, while fixing input, output, and auxiliary nodes. Each layerwise join and positionwise boundary edge is preserved under independent source and target actions.

Parameter graph. The node-bias encoding used in Section 4.1 has one node per neuron and one directed edge per weight:

$$\begin{aligned} U_\ell &= \{u_{\ell,i} : i \in [d_\ell]\}, \qquad V = \bigsqcup_{\ell=0}^{L} U_\ell, \\ E &= \bigsqcup_{\ell=1}^{L} \{e_{\ell,ij} : u_{\ell-1,j} \to u_{\ell,i} \mid i \in [d_\ell],\ j \in [d_{\ell-1}]\}. \end{aligned}$$

The parameter graph $\mathcal{P}_A(\boldsymbol{v}) = (V, E)$ carries scalar features

$$\omega_{\boldsymbol{v}}(e_{\ell,ij}) = (\boldsymbol{W}_\ell)_{ij}, \qquad \omega_{\boldsymbol{v}}(u_{\ell,i}) = \begin{cases} 0, & \ell = 0, \\ (\boldsymbol{b}_\ell)_i, & 1 \le \ell \le L. \end{cases}$$

For the classifier encoding, replace node-stored biases by edges $e^b_{\ell,i} : \eta_\ell \to u_{\ell,i}$ of value $(\boldsymbol{b}_\ell)_i$, adding one fixed auxiliary node η_ℓ per affine layer. Both encodings use the structural features of Definition 7; their feature dimensions and reciprocal message-passing edges are specified in Appendix D.1.

Joint graph. For source and target MLPs with $n = L_{\mathrm{src}} - 1$ and $m = L_{\mathrm{tgt}} - 1$ hidden layers, use the strided pairs $\mathcal{M}_{n,m}$ from Equation 8 and the positionwise boundary edges of Equation 7:

$$E^{\mathrm{MLP}}_\times = E_{\times,\partial} \cup \bigcup_{(j,i)\in\mathcal{M}_{n,m}} \mathrm{Join}(U^{\mathrm{src}}_{j+1}, U^{\mathrm{tgt}}_{i+1}). \tag{9}$$

Here $U^\alpha_{\mathrm{in}} = U^\alpha_0$ and $U^\alpha_{\mathrm{out}} = U^\alpha_{L_\alpha}$ for $\alpha \in \{\mathrm{src}, \mathrm{tgt}\}$. The joint graph is Equation 6 with $E_\times = E^{\mathrm{MLP}}_\times$; auxiliary bias nodes receive no cross edges. The all-to-all policy replaces $\mathcal{M}_{n,m}$ by $\{0, \dots, n-1\} \times \{0, \dots, m-1\}$.

B.2.3 Convolutional networks

Weight space. Let a convolutional stack have channel counts $\boldsymbol{c} = (c_0, \dots, c_L)$, kernel sizes $\boldsymbol{k} = (k_1, \dots, k_L)$, and pointwise nonlinearity σ. Its weight space is

$$\mathcal{V}_A = \bigoplus_{\ell=1}^{L} \left(\mathbb{R}^{c_\ell \times c_{\ell-1} \times k_\ell \times k_\ell} \oplus \mathbb{R}^{c_\ell}\right).$$

Realized function. For $\boldsymbol{v} = (K_1, \boldsymbol{b}_1, \dots, K_L, \boldsymbol{b}_L)$, define the affine convolution maps $\mathcal{C}_\ell(X) = K_\ell * X + \boldsymbol{b}_\ell$, where $*$ denotes spatial convolution with the architecture's fixed stride and padding, and biases are broadcast over spatial positions. The realized function is:

$$f_{\boldsymbol{v}}(\boldsymbol{x}) = \big(\mathcal{C}_L \circ \sigma \circ \mathcal{C}_{L-1} \circ \cdots \circ \sigma \circ \mathcal{C}_1\big)(\boldsymbol{x}).$$

Fixed channelwise pooling maps may be inserted between layers. Any dense layers following the convolutional stack use the MLP parameterization above.

Permutation symmetries. The hidden-channel group is $G_A = \prod_{\ell=1}^{L-1} S_{c_\ell}$. For $g = (\tau_1, \dots, \tau_{L-1})$, with τ_0 and τ_L the identity, $\rho_A(g)$ applies $P_{\tau_\ell}^\top$ to the output-channel axis and $P_{\tau_{\ell-1}}$ to the input-channel axis of the ℓ-th kernel, and sets $\boldsymbol{b}'_\ell = P_{\tau_\ell}^\top \boldsymbol{b}_\ell$. Channelwise nonlinearities and pooling commute with this action; flattening induces the corresponding permutation of the input coordinates of any following dense layer. Thus $f_{\rho_A(g)\boldsymbol{v}} = f_{\boldsymbol{v}}$. In the parameter graph, the same action relabels the hidden channel nodes and their incident kernel and bias edges.

Parameter graph. The corresponding parameter graph $\mathcal{P}_A(\boldsymbol{v}) = (V, E)$ has one node per channel and one auxiliary bias node per convolutional layer:

$$U_\ell = \{u_{\ell,i} : i \in [c_\ell]\}, \qquad V = \bigsqcup_{\ell=0}^{L} U_\ell \sqcup \{\eta_\ell : \ell \in [L]\},$$

Its edge set is

$$E = \bigsqcup_{\ell=1}^{L} \Big(\{e_{\ell,ijrs} : u_{\ell-1,j} \to u_{\ell,i} \mid i \in [c_\ell],\ j \in [c_{\ell-1}],\ r, s \in [k_\ell]\} \\ \sqcup \{e^b_{\ell,i} : \eta_\ell \to u_{\ell,i} \mid i \in [c_\ell]\}\Big),$$

with scalar edge values

$$\omega_{\boldsymbol{v}}(e_{\ell,ijrs}) = (K_\ell)_{ijrs}, \qquad \omega_{\boldsymbol{v}}(e^b_{\ell,i}) = (\boldsymbol{b}_\ell)_i.$$

Thus each input/output channel pair has k_ℓ^2 parallel kernel edges. Spatial reuse of a kernel entry does not create additional parameter edges (Lim et al., 2024, Section 2.1.2). Edge indices specify the parameter-to-edge correspondence; structural features are those of Definition 7. Channelwise activations and pooling reuse the channel nodes. If a dense readout follows the stack, continue the layer-node notation as $U_{L+1}, \dots, U_{L_{\mathrm{net}}}$ using the MLP construction, where L_{net} is the total number of convolutional and dense layers. For the stack alone, $L_{\mathrm{net}} = L$.

Joint graph for width compression. Our CNN experiments change widths only, so source and target have the same sequence of L_{net} layers. Using the node sets U_ℓ^{src} and U_ℓ^{tgt} of their parameter graphs, join corresponding hidden layers:

$$E_\times^{\mathrm{CNN}} = E_{\times,\partial} \cup \bigcup_{\ell=1}^{L_{\mathrm{net}}-1} \mathrm{Join}(U_\ell^{\mathrm{src}}, U_\ell^{\mathrm{tgt}}). \tag{10}$$

This is the equal-depth matching $\mathcal{M}_{n,n}$ of Equation 8, with $n = L_{\mathrm{net}} - 1$. The boundary edges $E_{\times,\partial}$ follow Equation 7, with $U_{\mathrm{in}}^\alpha = U_0^\alpha$ and $U_{\mathrm{out}}^\alpha = U_{L_{\mathrm{net}}}^\alpha$ for $\alpha \in \{\mathrm{src}, \mathrm{tgt}\}$. The CNN joint graph is Equation 6 with $E_\times = E_\times^{\mathrm{CNN}}$.

B.2.4 TRANSFORMERS

Weight space. Let a Transformer have L blocks, residual width d, H heads of dimension d_h, attention width $d_a = Hd_h$, and feed-forward width d_f. For blocks with biases and two affine LayerNorms, its weight space is

$$\mathcal{V}_A = \mathcal{V}_{\mathrm{emb}} \oplus \bigoplus_{\ell=0}^{L-1} \Big[\big(\mathbb{R}^{d_a \times d} \oplus \mathbb{R}^{d_a}\big)^{\oplus 3} \oplus \big(\mathbb{R}^{d \times d_a} \oplus \mathbb{R}^d\big) \\ \oplus \big(\mathbb{R}^{d_f \times d} \oplus \mathbb{R}^{d_f}\big) \oplus \big(\mathbb{R}^{d \times d_f} \oplus \mathbb{R}^d\big) \oplus \mathbb{R}^{4d} \Big] \oplus \mathcal{V}_{\mathrm{out}}.$$

The block factors contain the query, key, and value projections, the attention output projection, the two feed-forward maps, and the LayerNorm gains and shifts, respectively. The spaces $\mathcal{V}_{\mathrm{emb}}$ and $\mathcal{V}_{\mathrm{out}}$ contain the trainable input and positional embeddings (including any class token), and the output head and any final normalization, respectively.

Realized function. To specify the realized function, store tokens as columns of $X \in \mathbb{R}^{d \times n}$ and write $\mathbf{1}_n$ for the all-ones vector. Suppressing the block index, the projections for head h are

$$Q_h = W_{q,h}X + \boldsymbol{b}_{q,h}\mathbf{1}_n^\top, \qquad K_h = W_{k,h}X + \boldsymbol{b}_{k,h}\mathbf{1}_n^\top, \qquad V_h = W_{v,h}X + \boldsymbol{b}_{v,h}\mathbf{1}_n^\top.$$

Here $W_{q,h}, W_{k,h}, W_{v,h}$ are the d_h-row blocks of W_q, W_k, W_v. With softmax applied rowwise, define

$$T_h(X) = V_h \ \mathrm{softmax}\Big(Q_h^\top K_h / \sqrt{d_h}\Big)^\top,$$
$$\mathrm{MHA}(X) = W_o \ \mathrm{concat}_{h=1}^H T_h(X) + \boldsymbol{b}_o\mathbf{1}_n^\top,$$
$$\mathrm{FF}(X) = W_2\,\sigma\big(W_1X + \boldsymbol{b}_1\mathbf{1}_n^\top\big) + \boldsymbol{b}_2\mathbf{1}_n^\top,$$

where concatenation is along the feature dimension and σ acts entrywise. In the pre-normalization convention, block ℓ is

$$Y = X + \mathrm{MHA}_\ell(\mathrm{LN}_{\ell,1}(X)), \qquad B_\ell(X) = Y + \mathrm{FF}_\ell(\mathrm{LN}_{\ell,2}(Y)),$$

where each LayerNorm acts separately on each token, with learned gain and shift and a fixed positive stabilizer. The realized function is

$$f_{\boldsymbol{v}}(\boldsymbol{x}) = \big(R_{\boldsymbol{v}} \circ B_{L-1} \circ \cdots \circ B_0 \circ E_{\boldsymbol{v}}\big)(\boldsymbol{x}),$$

where $E_{\boldsymbol{v}}$ embeds the input and adds positional representations, and $R_{\boldsymbol{v}}$ applies the architecture's readout and any final normalization.

Permutation symmetries. We use permutations of coordinates within each head, permutations of heads, and permutations of feed-forward units, while fixing residual-stream coordinates:

$$G_A = \prod_{\ell=0}^{L-1} \left((S_{d_h} \wr S_H) \times S_{d_f} \right).$$

For (τ_ℓ, κ_ℓ) in the ℓ-th factor, the attention action applies $P_{\tau_\ell}^\top$ to the rows and biases of W_q, W_k, W_v and P_{τ_ℓ} to the columns of W_o; the feed-forward action applies $P_{\kappa_\ell}^\top$ to the rows and bias of W_1 and P_{κ_ℓ} to the columns of W_2. Joint query–key permutations preserve the attention logits, value permutations are absorbed by W_o, and feed-forward permutations are the usual one-hidden-layer MLP symmetry. Thus this action preserves the realized function.

This subgroup is compatible with the shared-coordinate attention graph of Lim et al. (2024, Section 2.1.2 and Figure 3). They describe attention edges as "a set that corresponds to the query, key, and value maps"; the shared intermediate nodes for these three projections are explicit in their released graph construction.[4] The subgroup does not exhaust the functional symmetries of Transformers. For example, standard dot-product attention also permits independent permutations of query–key and value coordinates within each head, as well as compensating invertible changes of basis in the query and key projections.

The symmetry implemented by a graph metanetwork is determined by the typed parameter graph, not only by the computational architecture. We therefore distinguish the chosen function-preserving group G_A above from

$$\widehat{G}_A := \{g \in G_A : \rho_A(g) \text{ is induced by a neural graph automorphism of } \mathcal{P}_A\}.$$

Whether $\widehat{G}_A = G_A$ depends on the structural labels preserved by the graph encoding; fixed head-index labels, for example, can restrict head permutations. Our implementation and all formal claims about the deployed Transformer operator use this represented subgroup $\widehat{G}_A$, without claiming equivariance to every functional symmetry.

Parameter graph. The parameter graph is built by joining the parameter subgraphs of these modules. For block ℓ, let

$$R_\ell^s = \{r_{\ell,i}^s : i \in [d]\} \quad (s = 0, 1, 2), \quad A_\ell = \{a_{\ell,p} : p \in [d_a]\}, \quad F_\ell = \{f_{\ell,j} : j \in [d_f]\}.$$

The sets $R_\ell^0, R_\ell^1, R_\ell^2$ index the residual coordinates at the block input, after attention, and at the block output. The nodes in A_ℓ jointly index query, key, and value coordinates, with $p = (h-1)d_h + r$ for coordinate r of head h; F_ℓ indexes the feed-forward hidden units. Add bias nodes $\mathcal{B}_\ell = \{\eta_{\ell,t} : t \in \{q, k, v, o, 1, 2\}\}$ and normalization nodes $\mathcal{N}_\ell = \{\nu_{\ell,s}^\gamma, \nu_{\ell,s}^\beta : s = 1, 2\}$. The block node set is

$$V_\ell = \Big(\bigsqcup_{s=0}^{2} R_\ell^s \Big) \sqcup A_\ell \sqcup F_\ell \sqcup \mathcal{B}_\ell \sqcup \mathcal{N}_\ell.$$

Writing $u \xrightarrow{\alpha} v$ for an edge whose scalar value is $\omega_{\boldsymbol{v}}(e) = \alpha$, the matrix edges are precisely

$$\begin{aligned}
r_{\ell,i}^0 &\xrightarrow{(W_{\ell,t})_{pi}} a_{\ell,p} \quad (t \in \{q, k, v\},\ i \in [d],\ p \in [d_a]),\\
a_{\ell,p} &\xrightarrow{(W_{\ell,o})_{ip}} r_{\ell,i}^1 \quad (p \in [d_a],\ i \in [d]),\\
r_{\ell,i}^1 &\xrightarrow{(W_{\ell,1})_{ji}} f_{\ell,j} \quad (i \in [d],\ j \in [d_f]),\\
f_{\ell,j} &\xrightarrow{(W_{\ell,2})_{ij}} r_{\ell,i}^2 \quad (j \in [d_f],\ i \in [d]).
\end{aligned}$$

In particular, each pair $(r_{\ell,i}^0, a_{\ell,p})$ has three parallel edges, one for each of W_q, W_k, W_v. For every affine map t, connect $\eta_{\ell,t}$ to each output-coordinate node of that map with the corresponding entry

[4] The function `self_attention_to_graph` uses the same `middle_neuron_idx` for the query, key, and value projections and the input of the output projection; see the released implementation.

of $\boldsymbol{b}_{\ell,t}$. LayerNorm s adds two edge families for its gain $\gamma_{\ell,s} \in \mathbb{R}^d$ and shift $\beta_{\ell,s} \in \mathbb{R}^d$:

$$\nu^{\gamma}_{\ell,s} \xrightarrow{(\gamma_{\ell,s})_i} r^{s-1}_{\ell,i}, \qquad \nu^{\beta}_{\ell,s} \xrightarrow{(\beta_{\ell,s})_i} r^{s-1}_{\ell,i} \qquad (i \in [d],\ s = 1, 2).$$

The remaining block edges are the parameter-free residual edges $r^0_{\ell,i} \xrightarrow{1} r^1_{\ell,i}$ and $r^1_{\ell,i} \xrightarrow{1} r^2_{\ell,i}$. These matrix, bias, normalization, and residual edges together define E_ℓ. As in Lim et al. (2024, Section 2.1.2 and Appendix B), activations, attention softmax, and tokenwise reuse of the projections introduce no additional parameter nodes or edges.

The full graph $\mathcal{P}_A(\boldsymbol{v}) = (V, E)$ is the union of these block graphs and the embedding and readout graphs, identifying $r^2_{\ell,i} = r^0_{\ell+1,i}$ at consecutive block boundaries. Patch embeddings use the convolutional construction above, and linear embeddings and readouts use the MLP edge rule. Denote their layer-node sets by U^{emb}_k and U^{out}_k, with depths L_{emb} and L_{out}, and identify

$$U^{\mathrm{emb}}_{L_{\mathrm{emb}}} = R^0_0, \qquad U^{\mathrm{out}}_0 = R^2_{L-1}.$$

The network's input and output node sets are $U_{\mathrm{in}} = U^{\mathrm{emb}}_0$ and $U_{\mathrm{out}} = U^{\mathrm{out}}_{L_{\mathrm{out}}}$. Any trainable positional or class-token vector is represented by an auxiliary node with one edge per entry to the corresponding embedding coordinate; final normalization uses the same gain/shift construction. Fixed positional representations introduce no learnable edges. The node indices above specify parameter locations; structural labels and any reciprocal edges used for message passing follow Definition 7 and Appendix D.1.

Joint graph. Apply the strided matching $\mathcal{M}_{n,m}$ of Equation 8 to whole Transformer blocks, with $n = L_{\mathrm{src}}$ and $m = L_{\mathrm{tgt}}$. Within each matched block pair, join corresponding unit sets:

$$E_{\times,\mathrm{blocks}} = \bigcup_{(j,i)\in\mathcal{M}_{n,m}} \Big[\bigcup_{s=0}^{2} \mathrm{Join}(R^{s,\mathrm{src}}_j, R^{s,\mathrm{tgt}}_i) \cup \mathrm{Join}(A^{\mathrm{src}}_j, A^{\mathrm{tgt}}_i) \cup \mathrm{Join}(F^{\mathrm{src}}_j, F^{\mathrm{tgt}}_i) \Big]. \tag{11}$$

Thus joins match attention coordinates, feed-forward hidden units, and residual coordinates at corresponding submodule boundaries. Attention joins include all coordinates on both sides, allowing different head counts.

For matching embedding and readout depths on the two sides, join their corresponding internal node sets $U^{t,\alpha}_k$ defined above, where $t \in \{\mathrm{emb}, \mathrm{out}\}$ and $\alpha \in \{\mathrm{src}, \mathrm{tgt}\}$. With the positionwise input/output edges of Equation 7, the full cross-edge set is

$$E^{\mathrm{Tr}}_{\times} = E_{\times,\partial} \cup E_{\times,\mathrm{blocks}} \cup \bigcup_{t\in\{\mathrm{emb},\mathrm{out}\}} \bigcup_{k=1}^{L_t-1} \mathrm{Join}(U^{t,\mathrm{src}}_k, U^{t,\mathrm{tgt}}_k),$$

which defines the Transformer joint graph through Equation 6. The embedding–block and block–readout interfaces are already included through $R^{0,\alpha}_0$ and $R^{2,\alpha}_{L_\alpha-1}$. For a single patch-embedding layer and linear classifier, $L_{\mathrm{emb}} = L_{\mathrm{out}} = 1$, so the last union is empty. Set unions include each cross edge once, even when joins overlap after the block-boundary identifications.

B.2.5 Equivariance of the joint graph

Proposition 2 (Equivariance induced by the joint graph)**.** *Let $\mathcal{J}$ be the joint graph of Equation 4, with the cross-edge construction of Section 4.1 (and Appendix B.2.2) for MLPs, Appendix B.2.3 for CNNs, or Appendix B.2.4 for Transformers. Let GMN_θ be a graph metanetwork as defined by Lim et al. (2024, Section 2.3), and let H_{src} and H_{tgt} be parameter-permutation groups induced by neural graph automorphisms of the respective parameter graphs. Then the CrossGMN operator of Definition 6 satisfies, for every choice of parameters θ,*

$$\mathrm{CrossGMN}_\theta(g_{\mathrm{src}} \cdot \boldsymbol{v}_{\mathrm{src}}, g_{\mathrm{tgt}} \cdot \boldsymbol{v}_{\mathrm{init}}) = g_{\mathrm{tgt}} \cdot \mathrm{CrossGMN}_\theta(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$$

for all $g_{\mathrm{src}} \in H_{\mathrm{src}}$ and $g_{\mathrm{tgt}} \in H_{\mathrm{tgt}}$ and all source and target parameters. For MLPs, we may take $H_{\mathrm{src}} = G_{\mathrm{src}}$ and $H_{\mathrm{tgt}} = G_{\mathrm{tgt}}$, so this is exactly the symmetry required by Definition 4.

Proof. Fix $(g_{\mathrm{src}}, g_{\mathrm{tgt}}) \in H_{\mathrm{src}} \times H_{\mathrm{tgt}}$ and write $\mathcal{J} = \mathcal{J}(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})$. Let γ be the disjoint union of inducing automorphisms of the two parameter graphs. Each layerwise join is preserved because its endpoint sets are preserved; any positionwise boundary edges are fixed. Cross-edge features are unchanged within each directed join. Thus γ preserves the joint graph's structure and structural features, and

$$\mathcal{J}(g_{\mathrm{src}} \cdot \boldsymbol{v}_{\mathrm{src}}, g_{\mathrm{tgt}} \cdot \boldsymbol{v}_{\mathrm{init}}) = \gamma \cdot \mathcal{J}.$$

Graph metanetworks commute with such relabelings (Lim et al., 2024, Appendix C.6), so $\mathrm{GMN}_\theta(\gamma \cdot \mathcal{J}) = \gamma \cdot \mathrm{GMN}_\theta(\mathcal{J})$. Moreover, the target projection satisfies $\Pi_{\mathrm{tgt}}(\gamma \cdot Z) = g_{\mathrm{tgt}} \cdot \Pi_{\mathrm{tgt}}(Z)$ for every joint output Z: it discards source entries and decodes target entries into their corresponding parameter coordinates. Consequently,

$$\begin{aligned}\boldsymbol{\Delta}_\theta(g_{\mathrm{src}} \cdot \boldsymbol{v}_{\mathrm{src}}, g_{\mathrm{tgt}} \cdot \boldsymbol{v}_{\mathrm{init}}) &= \Pi_{\mathrm{tgt}}(\mathrm{GMN}_\theta(\gamma \cdot \mathcal{J})) \\ &= \Pi_{\mathrm{tgt}}(\gamma \cdot \mathrm{GMN}_\theta(\mathcal{J})) \\ &= g_{\mathrm{tgt}} \cdot \boldsymbol{\Delta}_\theta(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}).\end{aligned}$$

Since g_{tgt} acts linearly on target parameters, adding the transformed initialization and applying the fixed residual scale s in Equation 5 gives the claimed identity. Taking $g_{\mathrm{tgt}} = e$ gives source invariance, and taking $g_{\mathrm{src}} = e$ gives target equivariance. □

C Theoretical Results

This appendix proves the three claims used in the main text. First, mapping each larger network to an optimal smaller one can be discontinuous (Appendix C.1). Second, any fixed number of standard distillation steps does define a continuous cross-architecture operator (Appendix C.2). Third, Cross-GMN is universal for continuous source-invariant, target-equivariant operators on compact general-position sets (Appendix C.3), including a finite collection of source architectures (Appendix C.4).

C.1 Discontinuity of mapping a larger network to an optimal smaller one

In model compression, mapping each larger network to an optimal smaller one can be discontinuous. Following Section 3, we call the larger source network the *teacher* and the smaller target network the *student*, with parameters $\boldsymbol{v}_T \in \mathcal{V}_T$ and $\boldsymbol{v}_S \in \mathcal{V}_S$. We consider compressing a ReLU MLP with one hidden layer from width 2 to width 1. The optimal student switches between the teacher's two spatially separated ReLU components as their coefficients cross. Figure 3 illustrates this discontinuity.

Formal construction. Take $A_T = ((1, 2, 1), \mathrm{ReLU})$ and $A_S = ((1, 1, 1), \mathrm{ReLU})$. On $I = [-11, 10]$, define

$$u(x) = \mathrm{ReLU}(-x - 10), \qquad v(x) = \mathrm{ReLU}(x - 9). \tag{12}$$

All norms and inner products below are on I. The supports of u and v are disjoint, and

$$\|u\|_2^2 = \|v\|_2^2 = \frac{1}{3}, \qquad \langle u, v\rangle = 0, \qquad \|u - v\|_\infty = 1. \tag{13}$$

For $t \in [-\frac{1}{2}, \frac{1}{2}]$, choose teacher parameters $\boldsymbol{v}_T(t)$ realizing

$$f_t := f_{\boldsymbol{v}_T(t)} = (1 - t)u + (1 + t)v.$$

An explicit continuous parameter path is

$$\boldsymbol{W}_{T,1} = \begin{bmatrix}-1\\ 1\end{bmatrix}, \quad \boldsymbol{b}_{T,1} = \begin{bmatrix}-10\\ -9\end{bmatrix}, \quad \boldsymbol{W}_{T,2}(t) = [1 - t \quad 1 + t], \quad \boldsymbol{b}_{T,2} = 0. \tag{14}$$

Restrict the student to zero output bias:

$$\mathcal{V}_S^0 = \{(w, b, a, 0) : w, b, a \in \mathbb{R}\} \subset \mathcal{V}_S, \qquad f_{\boldsymbol{v}_S}(x) = a\,\mathrm{ReLU}(wx + b).$$

Its realized function class is $\mathcal{S} = \{f_{\boldsymbol{v}_S}|_I : \boldsymbol{v}_S \in \mathcal{V}_S^0\}$, as in Figure 3. Use squared functional error,

$$L(\boldsymbol{v}_T(t), \boldsymbol{v}_S) = L_t(f_{\boldsymbol{v}_S}), \qquad L_t(g) := \int_I |f_t(x) - g(x)|^2\, dx, \tag{15}$$

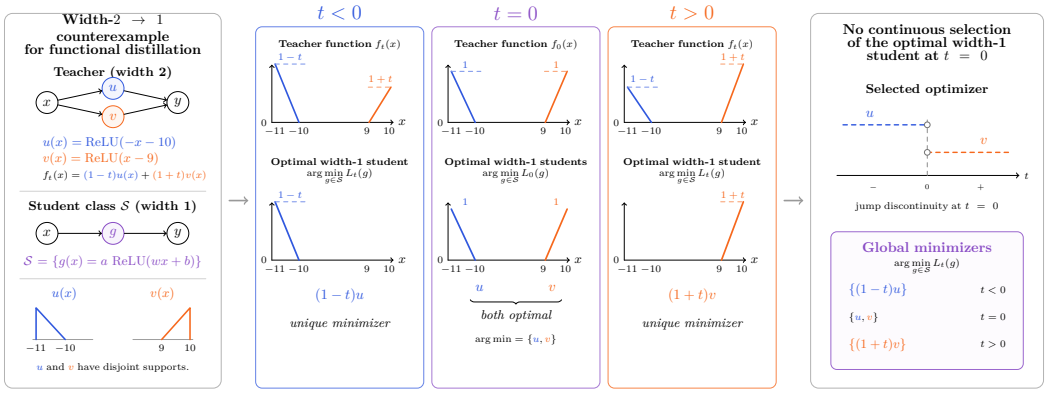


Figure 3: **Mapping a teacher to an optimal student can be discontinuous.** The width-2 teacher parameters $\boldsymbol{v}_T(t)$ in Equation 14 vary continuously and realize $f_t = (1-t)u + (1+t)v$. By Lemma 3, the optimal width-1 student realizes $(1-t)u$ for $t < 0$ and $(1+t)v$ for $t > 0$; at $t = 0$, both u and v are optimal. These one-sided limits differ, so no continuous parameter map can select an optimal student for every teacher along the path (Proposition 4).

which is the loss of Equation 3 for uniform inputs on I, up to a constant factor. The optimal student parameters are

$$\mathrm{Opt}(\boldsymbol{v}_T(t)) = \arg\min_{\boldsymbol{v}_S \in \mathcal{V}_S^0} L(\boldsymbol{v}_T(t), \boldsymbol{v}_S). \tag{16}$$

Lemma 3 (Best one-hinge approximation)**.** *The functions realized by the optimal students are*

$$\{f_{\boldsymbol{v}_S}|_I : \boldsymbol{v}_S \in \mathrm{Opt}(\boldsymbol{v}_T(t))\} = \begin{cases} \{(1-t)u\}, & t < 0, \\ \{u, v\}, & t = 0, \\ \{(1+t)v\}, & t > 0. \end{cases} \tag{17}$$

Proof. On I, every student in $\mathcal{V}_S^0$ realizes a scalar multiple of either the constant function, a right-facing hinge $h_c^+(x) = \mathrm{ReLU}(x - c)$, or a left-facing hinge $h_c^-(x) = \mathrm{ReLU}(c - x)$. For a nonzero candidate shape h, optimizing its scalar coefficient gives

$$\min_{\lambda \in \mathbb{R}} \|f_t - \lambda h\|_2^2 = \|f_t\|_2^2 - \Gamma_t(h), \qquad \Gamma_t(h) = \frac{\langle f_t, h\rangle^2}{\|h\|_2^2}. \tag{18}$$

Thus $\Gamma_t(h)$ is the reduction in squared error obtained by fitting the best scalar multiple of h, relative to the zero function. Minimizing the student's error is therefore equivalent to maximizing $\Gamma_t(h)$.

For the teacher's two ReLU components,

$$\Gamma_t(u) = \frac{(1-t)^2}{3}, \qquad \Gamma_t(v) = \frac{(1+t)^2}{3}.$$

For the constant function $h(x) \equiv 1$, we have $\int_I f_t(x)\,dx = (1-t)/2 + (1+t)/2 = 1$ and $\|h\|_2^2 = |I| = 21$. Hence

$$\Gamma_t(1) = \frac{1}{21} < \frac{1}{12} \le \min\{\Gamma_t(u), \Gamma_t(v)\},$$

where the last inequality follows from $|t| \le 1/2$. Thus the best constant student has strictly larger error than either optimally scaled component and cannot be optimal.

Consider a nonzero right-facing hinge h_c^+. If $c \ge -10$, it does not overlap the support of u, and Cauchy–Schwarz gives $\Gamma_t(h_c^+) \le \Gamma_t(v)$, with equality only when h_c^+ is proportional to v, i.e. only when $c = 9$. If $c < -10$, put $\tau = -10 - c > 0$ and $M_t = \max\{1-t, 1+t\}$. Direct integration gives

$$\|h_c^+\|_2^2 \ge \frac{8000 + 1200\tau + 60\tau^2}{3}, \qquad \langle f_t, h_c^+\rangle \le M_t\left(\frac{59}{6} + \tau\right),$$

and hence

$$\Gamma_t(h_c^+) \le \frac{3M_t^2(59/6+\tau)^2}{8000+1200\tau+60\tau^2} < \frac{M_t^2}{3}.$$

The last inequality is equivalent to $51\tau^2 + 1023\tau + 28519/4 > 0$. Reflecting the interval by $x \mapsto -1-x$ exchanges u and v and gives the same conclusion for every left-facing hinge other than u. Therefore the maximizing shape is u for $t<0$, v for $t>0$, and both for $t=0$. The minimizing scalar in equation 18 is respectively $1-t$ or $1+t$, proving equation 17. □

Proposition 4 (Discontinuity of mapping teachers to optimal students)**.** *Let $K_T = \{\boldsymbol{v}_T(t) : t \in [-\frac{1}{2}, \frac{1}{2}]\} \subset \mathcal{V}_T$. There is no continuous map $T : K_T \to \mathcal{V}_S$ such that $T(\boldsymbol{v}_T(t)) \in \mathrm{Opt}(\boldsymbol{v}_T(t))$ for every t. More strongly, for any continuous $T : K_T \to \mathcal{V}_S$ and any selection $T^\star : K_T \to \mathcal{V}_S^0$ with $T^\star(\boldsymbol{v}_T(t)) \in \mathrm{Opt}(\boldsymbol{v}_T(t))$,*

$$\sup_{t\in[-1/2,1/2]} \|f_{T(\boldsymbol{v}_T(t))} - f_{T^\star(\boldsymbol{v}_T(t))}\|_\infty \ge \frac{1}{2}. \tag{19}$$

Proof. The realization map $\boldsymbol{v}_S \mapsto f_{\boldsymbol{v}_S}|_I$ is continuous into $C(I)$ with the supremum norm. Hence a continuous T induces a continuous path $\varphi(t) = f_{T(\boldsymbol{v}_T(t))}|_I$. If the supremum in Equation 19 were $\varepsilon < 1/2$, then Lemma 3 and the limits $t \uparrow 0$ and $t \downarrow 0$ would give

$$\|\varphi(0) - u\|_\infty \le \varepsilon, \qquad \|\varphi(0) - v\|_\infty \le \varepsilon.$$

The triangle inequality would imply $1 = \|u - v\|_\infty \le 2\varepsilon < 1$, a contradiction. A continuous map selecting optimal students would have zero gap, so none exists. □

Remark 1 (The counterexample is in general position)**.** *General position, as defined in Equation 23, requires distinct biases within each hidden layer. Along the entire teacher path $\boldsymbol{v}_T(t)$, the hidden biases are -10 and -9, including at $t=0$. The width-1 student satisfies this condition vacuously. Hence $K_T \subset \mathcal{V}_T \setminus E_{A_T}$ and $E_{A_S} = \varnothing$. The approximation lower bound in Proposition 4 therefore holds even in general position: the obstruction is the discontinuous choice of an optimal student.*

Remark 2 (Refinement from a student initialization)**.** *The proposition concerns mapping each teacher to a globally optimal student. A cross-architecture operator $F(\boldsymbol{v}_T, \boldsymbol{v}_0)$ instead refines a supplied student initialization $\boldsymbol{v}_0$, without requiring global optimality. Appendix C.2 proves that a fixed number of distillation steps gives a continuous equivariant operator under the stated assumptions.*

C.2 Finite-step distillation is a cross-architecture operator

Recall from Section 3 that knowledge distillation trains a student with parameters $\boldsymbol{v}_S$ to match a teacher with parameters $\boldsymbol{v}_T$ by minimizing

$$L(\boldsymbol{v}_T, \boldsymbol{v}_S) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\ell(f_{\boldsymbol{v}_T}(\boldsymbol{x}), f_{\boldsymbol{v}_S}(\boldsymbol{x}))\right],$$

where ℓ measures the discrepancy between their outputs (Equation 3). Starting from a student initialization $\boldsymbol{v}_0$, this loss can be optimized with standard first-order methods, including gradient descent (GD) (Bottou et al., 2018), stochastic gradient descent (SGD) (Bottou, 2012), momentum (Polyak, 1964), RMSProp (Tieleman & Hinton, 2012), and Adam (Kingma & Ba, 2015). These methods operate *coordinatewise*: after computing the gradient, they apply the same scalar update rule to each parameter, using its own value and its current or past gradient components. The gradient itself may depend on all teacher and student parameters.

For full-batch optimization, set $L_j = L$ at every step. For SGD, fix the sequence of sampled minibatches and let L_j be the average distillation loss over the minibatch at step j; the same convention applies to minibatch versions of the other methods. Assume each L_j is differentiable in the student parameters and $\nabla_{\boldsymbol{v}_S} L_j(\boldsymbol{v}_T, \boldsymbol{v}_S)$ is jointly continuous in $(\boldsymbol{v}_T, \boldsymbol{v}_S)$. Write G_T and G_S for the teacher and student permutation groups, and $d_S = \dim \mathcal{V}_S$.

Proposition 5 (Finite-step distillation)**.** *Starting from $\boldsymbol{v}^{(0)} = \boldsymbol{v}_0$, suppose the optimizer has the coordinatewise form*

$$\begin{aligned} \boldsymbol{q}^{(j)} &= \nabla_{\boldsymbol{v}_S} L_j(\boldsymbol{v}_T, \boldsymbol{v}^{(j)}), \\ v_i^{(j+1)} &= u_j\big((\boldsymbol{v}_0)_i, q_i^{(0)}, \ldots, q_i^{(j)}\big), \qquad 1 \le i \le d_S, \end{aligned} \tag{20}$$

where each $u_j : \mathbb{R}^{j+2} \to \mathbb{R}$ is a fixed continuous function shared across all coordinates. Then, for every fixed integer $k \geq 0$, the map $\Phi_k(\boldsymbol{v}_T, \boldsymbol{v}_0) := \boldsymbol{v}^{(k)}$ is continuous and satisfies

$$\Phi_k(g_T \cdot \boldsymbol{v}_T, g_S \cdot \boldsymbol{v}_0) = g_S \cdot \Phi_k(\boldsymbol{v}_T, \boldsymbol{v}_0) \qquad (g_T \in G_T,\ g_S \in G_S).$$

Thus Φ_k is a continuous equivariant cross-architecture operator in the sense of Definition 4. Functionally equivalent teachers produce identical trajectories from the same initialization.

Proof. Each L_j depends only on the realized teacher and student functions, so

$$f_{\boldsymbol{v}_T} = f_{\boldsymbol{v}'_T} \Longrightarrow L_j(\boldsymbol{v}_T, \cdot) = L_j(\boldsymbol{v}'_T, \cdot), \qquad L_j(\boldsymbol{v}_T, P_g \boldsymbol{v}_S) = L_j(\boldsymbol{v}_T, \boldsymbol{v}_S), \tag{21}$$

where P_g is the permutation matrix of $g \in G_S$. Differentiating the second identity and using $P_g P_g^\top = I$ gives

$$\nabla_{\boldsymbol{v}_S} L_j(\boldsymbol{v}_T, P_g \boldsymbol{v}_S) = P_g \nabla_{\boldsymbol{v}_S} L_j(\boldsymbol{v}_T, \boldsymbol{v}_S). \tag{22}$$

Differentiating $L_j(\boldsymbol{v}_T, \cdot) = L_j(\boldsymbol{v}'_T, \cdot)$ likewise shows that functionally equivalent teachers give identical student gradients.

Continuity follows by induction: $\boldsymbol{v}^{(0)} = \boldsymbol{v}_0$ is continuous in $(\boldsymbol{v}_T, \boldsymbol{v}_0)$, and the jointly continuous gradients and continuous u_j make each subsequent iterate continuous by composition.

For symmetry, compare runs initialized at $(\boldsymbol{v}_T, \boldsymbol{v}_0)$ and $(\boldsymbol{v}'_T, P_g \boldsymbol{v}_0)$, where $f_{\boldsymbol{v}'_T} = f_{\boldsymbol{v}_T}$. Denote their student parameters at step r by $\boldsymbol{v}^{(r)}$ and $\widetilde{\boldsymbol{v}}^{(r)}$, respectively. Their student-loss gradient vectors are

$$\boldsymbol{q}^{(r)} = \nabla_{\boldsymbol{v}_S} L_r(\boldsymbol{v}_T, \boldsymbol{v}^{(r)}), \qquad \widetilde{\boldsymbol{q}}^{(r)} = \nabla_{\boldsymbol{v}_S} L_r(\boldsymbol{v}'_T, \widetilde{\boldsymbol{v}}^{(r)}).$$

Assume inductively that $\widetilde{\boldsymbol{v}}^{(r)} = P_g \boldsymbol{v}^{(r)}$ for $0 \leq r \leq j$. Substituting this relation, replacing the teacher using Equation 21, and applying the gradient transformation in Equation 22 gives

$$\begin{aligned} \widetilde{\boldsymbol{q}}^{(r)} &= \nabla_{\boldsymbol{v}_S} L_r(\boldsymbol{v}'_T, P_g \boldsymbol{v}^{(r)}) \\ &= \nabla_{\boldsymbol{v}_S} L_r(\boldsymbol{v}_T, P_g \boldsymbol{v}^{(r)}) \\ &= P_g \nabla_{\boldsymbol{v}_S} L_r(\boldsymbol{v}_T, \boldsymbol{v}^{(r)}) = P_g \boldsymbol{q}^{(r)} \qquad (0 \leq r \leq j). \end{aligned}$$

Writing $(P_g x)_i = x_{\pi_g(i)}$, the shared update rule in Equation 20 therefore yields

$$\begin{aligned} \widetilde{v}_i^{(j+1)} &= u_j\big((P_g \boldsymbol{v}_0)_i, \widetilde{q}_i^{(0)}, \ldots, \widetilde{q}_i^{(j)}\big) \\ &= u_j\big((\boldsymbol{v}_0)_{\pi_g(i)}, q_{\pi_g(i)}^{(0)}, \ldots, q_{\pi_g(i)}^{(j)}\big) = v_{\pi_g(i)}^{(j+1)}. \end{aligned}$$

Since $\widetilde{\boldsymbol{v}}^{(0)} = P_g \boldsymbol{v}_0$, induction gives $\widetilde{\boldsymbol{v}}^{(k)} = P_g \boldsymbol{v}^{(k)}$ for every fixed k. Taking $\boldsymbol{v}'_T = g_T \cdot \boldsymbol{v}_T$ proves the claimed equivariance, and taking g to be the identity proves equality of trajectories for functionally equivalent teachers. □

Lemma 6 (Standard gradient methods)**.** *Gradient descent, SGD, momentum, RMSProp, and Adam have the form in equation 20, with fixed scalar hyperparameter schedules, decay factors in $[0, 1)$, the usual zero initialization of running averages, and $\epsilon > 0$ in the denominators of RMSProp and Adam.*

Proof. Gradient descent and SGD subtract a scalar multiple of the current gradient. Unrolling momentum expresses each update as a weighted sum of the gradients computed so far. Likewise, the running averages in RMSProp and Adam are weighted sums of gradients or squared gradients, with coefficients independent of the parameter coordinate. Their remaining operations are coordinatewise square roots and division; Adam's bias corrections are fixed scalars at each step. All these operations are continuous, since the squared-gradient averages are nonnegative and $\epsilon > 0$ keeps denominators positive. Unrolling the parameter updates therefore gives a continuous function u_j of the initial coordinate and its gradient history, identical across coordinates, as required by equation 20. □

C.3 Universal approximation of cross-architecture operators

We now prove Theorem 1. The proof has two conceptual ingredients. First, every continuous source-invariant, target-equivariant map can be approximated by a finite sum of a source-invariant scalar times a target-equivariant operator. Second, CrossGMN can realize such finite mixtures: it computes the two factors on their respective parameter graphs, communicates the scalar through a cross edge, and multiplies them on the target side.

General position and the fixed-architecture theorem. For an MLP architecture $A = (\boldsymbol{d}, \sigma)$, let

$$E_A = \{\boldsymbol{v} : \text{two neurons in some hidden layer have equal biases}\}. \tag{23}$$

The complement is the general-position set used by Dayan et al. (2026). We use the following result.

Theorem 7 (Fixed-architecture universality; Dayan et al. (2026))**.** *On every compact $K \subset \mathcal{V}_A \setminus E_A$, GMNs uniformly approximate every continuous G_A-invariant functional and every continuous G_A-equivariant operator $K \to \mathcal{V}_A$.*

We may assume that each compact set contains every hidden-neuron permutation of each of its elements. To obtain this property, enlarge the sets to

$$\widetilde{K}_\alpha = \{g\cdot \boldsymbol{v} : \boldsymbol{v} \in K_\alpha,\ g \in G_\alpha\}, \qquad \alpha \in \{\mathrm{src}, \mathrm{tgt}\}.$$

Each enlarged set is a finite union of permuted copies of a compact set, so it remains compact. It also remains in general position, since permuting neurons only reorders the distinct biases within each hidden layer. Extend F to the enlarged sets by

$$\widetilde{F}(g_{\mathrm{src}}\cdot \boldsymbol{v}_{\mathrm{src}}, g_{\mathrm{tgt}}\cdot \boldsymbol{v}_{\mathrm{init}}) = g_{\mathrm{tgt}}\cdot F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}).$$

The symmetry condition in Equation 2 ensures that different ways of writing the same permuted input pair give the same output. The extension is continuous on each permuted copy of $K_{\mathrm{src}} \times K_{\mathrm{tgt}}$; since these are finitely many closed sets and the definitions agree on their overlaps, it is continuous on their union. Uniform approximation on the enlarged sets implies the same guarantee on the original sets. We therefore use the enlarged sets below, retaining the notation $K_{\mathrm{src}}, K_{\mathrm{tgt}}$, and F.

Lemma 8 (Invariant–equivariant finite mixture)**.** *Let $F : K_{\mathrm{src}} \times K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$ be continuous, source invariant, and target equivariant. For every $\eta > 0$, there are continuous source-invariant functions $a_1, \dots, a_M : K_{\mathrm{src}} \to [0,1]$ with $\sum_i a_i \equiv 1$ and continuous target-equivariant maps $h_1, \dots, h_M : K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$ such that*

$$\sup_{(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})} \left\| F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) - \sum_{i=1}^{M} a_i(\boldsymbol{v}_{\mathrm{src}}) h_i(\boldsymbol{v}_{\mathrm{init}}) \right\| < \eta. \tag{24}$$

Proof. Let $Q = K_{\mathrm{src}}/G_{\mathrm{src}}$ be the source orbit space. Because the group is finite and acts by isometries,

$$d_Q([\boldsymbol{v}], [\boldsymbol{v}']) = \min_{g \in G_{\mathrm{src}}} \|g\cdot \boldsymbol{v} - \boldsymbol{v}'\|$$

makes Q a compact metric space. Source invariance lets F factor through a continuous map $\overline{F} : Q \times K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$, and every $\overline{F}(q, \cdot)$ remains target equivariant. Uniform continuity implies that $q \mapsto \overline{F}(q, \cdot)$ is continuous in the supremum norm. Choose a finite cover $U_1, \dots, U_M$ of Q and points $q_i \in U_i$ such that

$$q \in U_i \quad \Longrightarrow \quad \|\overline{F}(q, \cdot) - \overline{F}(q_i, \cdot)\|_\infty < \eta.$$

Let $\{\alpha_i\}$ be a continuous partition of unity subordinate to this cover, set $a_i(\boldsymbol{v}) = \alpha_i([\boldsymbol{v}])$, and set $h_i(\boldsymbol{v}_{\mathrm{init}}) = \overline{F}(q_i, \boldsymbol{v}_{\mathrm{init}})$. Then the a_i are invariant, the h_i are equivariant, and the convex-combination bound gives equation 24. □

Lemma 9 (CrossGMN realizes finite mixtures)**.** *Let K_{src} and K_{tgt} be compact general-position sets for fixed source and target MLP architectures. For $i = 1, \dots, M$, let $a_i : K_{\mathrm{src}} \to \mathbb{R}$ be a continuous source-invariant functional, and let $h_i : K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$ be continuous and target equivariant. For*

every $\varepsilon > 0$, there exists a CrossGMN$_\theta$ whose joint graph uses the positionwise boundary joins of Section 4.1 and the strided cross-edge policy of Definition 5 such that

$$\sup_{(\boldsymbol{v}_{\mathrm{src}},\boldsymbol{v}_{\mathrm{init}})\in K_{\mathrm{src}}\times K_{\mathrm{tgt}}} \left\| \mathit{CrossGMN}_\theta(\boldsymbol{v}_{\mathrm{src}},\boldsymbol{v}_{\mathrm{init}}) - \sum_{i=1}^{M} a_i(\boldsymbol{v}_{\mathrm{src}})h_i(\boldsymbol{v}_{\mathrm{init}}) \right\| < \varepsilon. \tag{25}$$

The same conclusion holds if further symmetry-preserving cross edges are added.

Proof. Approximate the factors. Let $\boldsymbol{e}_{\mathrm{out}} \in \mathcal{V}_{\mathrm{src}}$ have value 1 in the first output-bias coordinate and 0 elsewhere, and define $\Psi_i(\boldsymbol{v}_{\mathrm{src}}) = a_i(\boldsymbol{v}_{\mathrm{src}})\boldsymbol{e}_{\mathrm{out}}$. This map is continuous and source equivariant because a_i is invariant and hidden-neuron permutations fix $\boldsymbol{e}_{\mathrm{out}}$. For any $\eta > 0$, Theorem 7 gives source and target GMNs S_i, H_i satisfying

$$\sup_{K_{\mathrm{src}}} \|S_i - \Psi_i\| < \eta, \qquad \sup_{K_{\mathrm{tgt}}} \|H_i - h_i\| < \eta.$$

Store and compute the factors in node and edge features. Let x_v, x_{uv} be the original node and edge features, including the parameter values and structural labels. At step j, use

$$z_v^{(j)} = \big(x_v, (w_{v,i}^{(j)})_{i=1}^M, c_v^{(j)}\big), \qquad z_{uv}^{(j)} = \big(x_{uv}, (w_{uv,i}^{(j)})_{i=1}^M, c_{uv}^{(j)}\big).$$

The block $w_{\cdot,i}$ stores the working state of S_i on the source side and of H_i on the target side; add zero coordinates and identity layers to match their dimensions and depths. The vectors $c_v, c_{uv} \in \mathbb{R}^M$ will store the source scalars and are initially zero. Keep x_v, x_{uv} unchanged throughout.

Set $\chi_\alpha(v) = \mathbf{1}_{\{v\in V^\alpha\}}$ and $\chi_\alpha(u,v) = \chi_\alpha(u)\chi_\alpha(v)$, where $\alpha \in \{\mathrm{src}, \mathrm{tgt}\}$. Initialize each working block with its side's GMN encoder applied to x. For block i, suppress i and the step index on its message, node-update, and edge-update maps $\phi^\alpha, U^\alpha, E^\alpha$, and use

$$\begin{aligned}
m_{uv,i}^{(j)} &= \sum_\alpha \chi_\alpha(u,v)\, \phi^\alpha(w_{u,i}^{(j)}, w_{v,i}^{(j)}, w_{uv,i}^{(j)}),\\
w_{v,i}^{(j+1)} &= \sum_\alpha \chi_\alpha(v)\, U^\alpha\left(w_{v,i}^{(j)}, \sum_{u:u\to v} m_{uv,i}^{(j)}\right),\\
w_{uv,i}^{(j+1)} &= \sum_\alpha \chi_\alpha(u,v)\, E^\alpha(w_{u,i}^{(j)}, w_{v,i}^{(j)}, w_{uv,i}^{(j)}).
\end{aligned}$$

Thus each side performs its own GMN computation, and cross edges send zero messages in this stage. Apply the local readouts and replace the working blocks by their scalar outputs. Let $\lambda(p)$ be the node or edge storing target parameter p, as in Appendix B.2.2, and let o_{src} be the first source output node. The resulting features contain

$$w_{o_{\mathrm{src}},i} = \widehat{a}_i := [S_i(\boldsymbol{v}_{\mathrm{src}})]_{\mathrm{out}}, \qquad w_{\lambda(p),i} = [\widehat{h}_i]_p, \quad \widehat{h}_i := H_i(\boldsymbol{v}_{\mathrm{init}}).$$

Here $[\cdot]_{\mathrm{out}}$ selects the first output-bias coordinate. Consequently $|\widehat{a}_i - a_i| < \eta$ and $\|\widehat{h}_i - h_i\| < \eta$ uniformly on the corresponding input sets.

Copy the source scalars to the target features. Write $\widehat{\boldsymbol{a}} = (\widehat{a}_1, \dots, \widehat{a}_M)$. On the positionwise cross edge from o_{src} to the first target output node o_{tgt}, send the message $(w_{o_{\mathrm{src}},i})_{i=1}^M = \widehat{\boldsymbol{a}}$ to the c field. Set all other messages in this step to zero, so that

$$c_v^{(0)} = \mathbf{1}_{\{v=o_{\mathrm{tgt}}\}}\widehat{\boldsymbol{a}} \qquad (v \in V^{\mathrm{tgt}}).$$

We now update only c, keeping the computed factors and original features fixed. Use the bidirectional internal message passing of Appendix D.1. Let $d(v)$ be the distance from o_{tgt} in the target graph, and let $N_-(v)$ consist of its target neighbors at distance $d(v)-1$. For $v \neq o_{\mathrm{tgt}}$, this set is nonempty. Apply

$$c_v^{(j+1)} = \begin{cases} c_v^{(j)}, & v = o_{\mathrm{tgt}},\\ \dfrac{1}{|N_-(v)|}\displaystyle\sum_{u\in N_-(v)} c_u^{(j)}, & v \neq o_{\mathrm{tgt}}. \end{cases}$$

The required neighbor masks and normalization constants are determined by the fixed MLP architecture, layer labels, and canonical output indices. Induction on $d(v)$ gives $c_v^{(j)} = \widehat{\boldsymbol{a}}$ for $j \geq d(v)$. After $D = \max_v d(v)$ steps, set $c_{uv} = c_v$ on each target edge. Every target parameter location now has the features

$$z_{\lambda(p)} = \big(x_{\lambda(p)}, ([\widehat{h}_i]_p)_{i=1}^M, (\widehat{a}_i)_{i=1}^M\big).$$

Multiply the stored factors and read out the parameters. Multiplication is continuous on $[-R, R]^2$. Hence, by the classical universal approximation theorems (Cybenko, 1989; Hornik, 1991), for every $R > 0$ and $\kappa > 0$ there is an MLP $\mu_{R,\kappa} : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\sup_{(x,y)\in[-R,R]^2} \big|\mu_{R,\kappa}(x, y) - xy\big| < \kappa. \tag{26}$$

Let $B = \max_i\{\|a_i\|_\infty, \|h_i\|_\infty\}$ and $P = \dim \mathcal{V}_{\mathrm{tgt}}$. Choose

$$0 < \eta < \min\left\{1, \frac{\varepsilon}{3M(2B+1)}\right\}, \qquad 0 < \kappa < \frac{\varepsilon}{3M\sqrt{P}}, \qquad R = B + 1.$$

Then $|\widehat{a}_i|, |[\widehat{h}_i]_p| < R$. Using the values stored at $\lambda(p)$, define the local readout

$$[\widehat{F}_\kappa]_p = \sum_{i=1}^M \mu_{R,\kappa}(c_{\lambda(p),i}, w_{\lambda(p),i}), \qquad [\boldsymbol{\Delta}]_p = \frac{[\widehat{F}_\kappa]_p - [\boldsymbol{v}_{\mathrm{init}}]_p}{s}.$$

The initialization coordinate is available in $x_{\lambda(p)}$. Thus the residual update of Definition 6 returns $\widehat{F}_\kappa$. In the Euclidean norm, uniformly on the input sets,

$$\begin{aligned}\left\|\widehat{F}_\kappa - \sum_i a_i h_i\right\| &\leq M\sqrt{P}\,\kappa + \sum_i \Big(|\widehat{a}_i - a_i|\,\|h_i\| + |\widehat{a}_i|\,\|\widehat{h}_i - h_i\|\Big) \\ &\leq M\sqrt{P}\,\kappa + M\eta(2B + \eta) < \frac{2\varepsilon}{3}.\end{aligned}$$

All local rules above are continuous on compact neighborhoods of the feature ranges, and their structural masks preserve the source and target permutations. Uniform MLP approximation of these rules and uniform continuity of their finite composition give a choice of θ such that

$$\sup_{K_{\mathrm{src}}\times K_{\mathrm{tgt}}} \left\|\mathrm{CrossGMN}_\theta - \widehat{F}_\kappa\right\| < \frac{\varepsilon}{3}.$$

The triangle inequality proves Equation 25. Additional cross edges are assigned zero messages in this construction. □

Theorem 1 (Universality of CrossGMN, restated)**.** *Let A_{src} and A_{tgt} be fixed MLP architectures, and let*

$$K_{\mathrm{src}} \subset \mathcal{V}_{A_{\mathrm{src}}} \setminus E_{A_{\mathrm{src}}}, \qquad K_{\mathrm{tgt}} \subset \mathcal{V}_{A_{\mathrm{tgt}}} \setminus E_{A_{\mathrm{tgt}}}$$

be compact sets, where E_A is defined in Equation 23. Let $F : K_{\mathrm{src}} \times K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$ be a continuous cross-architecture operator in the sense of Definition 4. For joint graphs containing the position-wise boundary joins and strided hidden-layer joins of Section 4.1, and every $\varepsilon > 0$, there exists a CrossGMN$_\theta$ such that

$$\sup_{(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})\in K_{\mathrm{src}}\times K_{\mathrm{tgt}}} \|\mathit{CrossGMN}_\theta(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}}) - F(\boldsymbol{v}_{\mathrm{src}}, \boldsymbol{v}_{\mathrm{init}})\| < \varepsilon.$$

Proof of Theorem 1. Apply Lemma 8 with error $\varepsilon/2$ and write $F_{\mathrm{mix}} = \sum_i a_i h_i$. Lemma 9 gives a CrossGMN within $\varepsilon/2$ of F_{mix}. The triangle inequality with equation 24 gives the desired uniform ε approximation of F.

C.4 One CrossGMN for finitely many source architectures

The same argument applies when the source population contains finitely many architectures and all map to one target architecture. The only additional requirement is that the source architecture be identifiable from the graph.

Definition 8 (Architecture-identifiable family)**.** A finite family $A_{\mathrm{src}}^{(1)}, \dots, A_{\mathrm{src}}^{(m)}$ is *architecture identifiable* if a finite message-passing block can write the one-hot architecture tag e_r on every node of a joint graph whose source has architecture $A_{\mathrm{src}}^{(r)}$. An explicit family feature is sufficient. For finite MLP families, graph depth, layer types, and layer cardinalities are also sufficient.

Theorem 10 (Finite heterogeneous source families)**.** *Let A_{tgt} be a fixed MLP architecture and let $A_{\mathrm{src}}^{(1)}, \dots, A_{\mathrm{src}}^{(m)}$ be an architecture-identifiable finite family of MLP architectures. Let*

$$K_r \subset \mathcal{V}_{A_{\mathrm{src}}^{(r)}} \setminus E_{A_{\mathrm{src}}^{(r)}} \quad (r = 1, \dots, m), \qquad K_{\mathrm{tgt}} \subset \mathcal{V}_{A_{\mathrm{tgt}}} \setminus E_{A_{\mathrm{tgt}}}$$

be compact sets, with E_A as in Equation 23. For each r, let $F_r : K_r \times K_{\mathrm{tgt}} \to \mathcal{V}_{\mathrm{tgt}}$ be continuous, source invariant, and target equivariant. Then one CrossGMN, with one parameter set, can approximate all F_r uniformly to any prescribed accuracy.

Proof. Fix $\varepsilon > 0$. For each r, Theorem 1 gives $\widehat{F}_r = \mathrm{CrossGMN}_{\theta_r}$ with

$$\sup_{K_r \times K_{\mathrm{tgt}}} \|\widehat{F}_r - F_r\| < \frac{\varepsilon}{2}.$$

Add zero hidden coordinates and identity layers so that these networks have common hidden-state dimensions and depth.

Let x_v and x_{uv} denote the original node and edge features, including their parameter values and structural labels. On a joint graph with source architecture $A_{\mathrm{src}}^{(r)}$, the identification block of Definition 8 writes the tag $e_r \in \mathbb{R}^m$ on every source and target node; its r-th entry is 1 and all other entries are 0. Copy this tag to each edge from either endpoint. At step j, use node and edge features

$$z_v^{(j)} = (x_v, e_r, h_v^{(j)}), \qquad z_{uv}^{(j)} = (x_{uv}, e_r, h_{uv}^{(j)}),$$

where $h_v^{(j)}$ and $h_{uv}^{(j)}$ are the working hidden states. Initialize these states to zero. The original features and tags remain unchanged throughout; the first encoding step fills the working states.

For each encoding, message, node-update, edge-update, or readout operation j, let $\Phi_j^{(q)}$ be the corresponding rule of network q, with inputs padded to common dimensions. If τ is the stored tag and z collects the operation's other inputs, use the shared rule

$$\Phi_j(\tau, z) = \sum_{q=1}^{m} \tau_q \Phi_j^{(q)}(z).$$

A message uses the tag at its receiving node; node and edge operations use their own stored tags. Since all tags equal e_r, $\Phi_j(e_r, z) = \Phi_j^{(r)}(z)$ at every operation. Induction over the encoding, message-passing steps, and readout therefore shows that these rules return exactly $\widehat{F}_r$ on $K_r \times K_{\mathrm{tgt}}$.

The rules Φ_j are continuous and can be uniformly approximated by MLPs on compact neighborhoods of the local feature ranges (Cybenko, 1989; Hornik, 1991). There are finitely many rules, layers, and graph sizes, and each input set is compact. Uniform continuity of the resulting finite compositions therefore lets us choose their approximation accuracies so that one assembled CrossGMN satisfies

$$\max_{1 \le r \le m} \sup_{K_r \times K_{\mathrm{tgt}}} \|\mathrm{CrossGMN}_\theta - \widehat{F}_r\| < \frac{\varepsilon}{2}.$$

The triangle inequality gives

$$\max_{1 \le r \le m} \sup_{K_r \times K_{\mathrm{tgt}}} \|\mathrm{CrossGMN}_\theta - F_r\| < \varepsilon.$$

All local rules are shared across architectures, so θ is a single parameter set, as required. □

C.5 Scope for CNN and Transformer weight spaces

Proposition 2 establishes equivariance for CNNs and Transformers directly from their implemented parameter graphs. The universality proof extends to these families as long as the fixed-architecture invariant and equivariant universality for GMNs on their parameter graphs still hold. The extension applies with respect to the neural graph automorphism groups defined in Appendix B.2.

D Experimental Details

Throughout, *CrossGMN* denotes the operator in Equation 5, and the *anchor* is the target initialization $\boldsymbol{v}_0$ that it refines. As in Section 5.1, the source network is the teacher $\boldsymbol{v}_T$ and the target network is the student, whose predicted initialization is $\hat{\boldsymbol{v}}_S$.

D.1 CrossGMN implementation

All experiments use the joint graph of the teacher $\boldsymbol{v}_T$ and anchor $\boldsymbol{v}_0$. An equivariant graph metanetwork processes this graph, and the target-side readout predicts the residual in Equation 5.

In the classifier experiments, node features concatenate a scalar, an 8-dimensional layer embedding, a 4-dimensional source/target-role embedding, a 4-dimensional node-type embedding, and optional 4-dimensional family and activation embeddings. Edge features concatenate the parameter value, the layer embedding, a 4-dimensional edge-role embedding, and an optional family embedding. In the INR experiments, node and edge inputs both have dimension $1 + 4 + 4 = 9$: scalar value, layer embedding, and role embedding. Parameter values are not normalized. The four edge roles are source-internal, target-internal, source-to-target, and target-to-source.

Cross edges. Cross edges use the layerwise joins and matching notation of Appendix B.2.1. The formal joint graphs in Appendices B.2.2–B.2.4 use positionwise input/output joins. The CNN and ViT classifier encoding uses dense boundary joins, replacing Pos by Join in Equation 7; MLP boundaries remain positionwise. Both choices preserve the stated symmetries because input and output nodes are fixed.

Backbone and readout. We use a bidirectional ScaleGMN backbone (Kalogeropoulos et al., 2024), motivated by its better empirical performance than plain GMN in most experiments. Our universality results are established for plain GMN. The backbone uses skip connections, edge updates, and dropout 0.1 during operator training. Message and update functions are two-layer SiLU MLPs with layer normalization. The classifier families use scale-equivariant message passing; the SIREN families use sign-equivariant message passing, except for the parameter-matched permutation-equivariant ModelNet40 width+depth operator. Table 8 gives the configuration for each model family.

Table 8: CrossGMN configurations. d_v/d_e are node/edge input widths and $|\theta|$ is the metanetwork parameter count; s is the residual scale in Equation 5.

Family / operator group	Symmetry	d_{hid}	Layers	Agg.	d_v/d_e	$\lvert\theta\rvert$	s
MNIST SIREN (all; depth and width+depth use $s = .01$)	sign	128	4	mean	$9/9$	2,647,895	.005
ModelNet40 depth $4 \to 2$	sign	192	4	add	$9/9$	5,937,947	.02
ModelNet40 depth $4 \to 1$	sign	192	4	add	$9/9$	5,937,907	.02
ModelNet40 width	sign	192	4	add	$9/9$	5,937,907	.005
ModelNet40 width+depth	permutation	324	4	add	$9/9$	6,010,354	.01
ViT depth	scale	64	4	mean	$25/17$	405,587	.1
ViT attention dimension, head count, head count+depth	scale	128	6	mean	$25/17$	2,438,163	.1
CNN width	scale	64	6	mean	$21/13$	620,607	.4
MLP width / depth / width+depth	scale	128	8	mean	$17/13$	3,293,663	.1
MLP heterogeneous architectures	scale	128	4	mean	$17/13$	1,579,487	.1

Operator objective. We train CrossGMN directly on teacher–student outputs, without an inner distillation loop. For a minibatch of B teachers, write

$$\hat{\boldsymbol{v}}_{S,i} = \mathrm{CrossGMN}_\theta(\boldsymbol{v}_{T,i}, \boldsymbol{v}_{0,i}) = \boldsymbol{v}_{0,i} + s\,\boldsymbol{\Delta}_\theta(\boldsymbol{v}_{T,i}, \boldsymbol{v}_{0,i}), \qquad i = 1, \dots, B,$$

where $\boldsymbol{v}_{0,i}$ is the corresponding anchor. For classifiers, a shared image batch X gives the objective

$$\mathcal{L}(\theta) = \frac{1}{B|X|} \sum_{i=1}^{B} \sum_{\boldsymbol{x} \in X} \mathrm{KL}\big(\mathrm{softmax}(f_{\boldsymbol{v}_{T,i}}(\boldsymbol{x})) \,\big\|\, \mathrm{softmax}(f_{\hat{\boldsymbol{v}}_{S,i}}(\boldsymbol{x}))\big) . \tag{27}$$

The temperature is 1; operator training uses neither a cross-entropy term nor logit clipping.

For INRs, we minimize the MSE between teacher and predicted-student outputs. Let $\mathcal{G} \subset \mathbb{R}^2$ be the full 28×28 pixel grid and $\mathcal{P}_i \subset \mathbb{R}^3$ the fixed set of uniformly sampled query points for ModelNet40 teacher i. The respective objectives are

$$\mathcal{L}_{\mathrm{render}}(\theta) = \frac{1}{B|\mathcal{G}|} \sum_{i=1}^{B} \sum_{\boldsymbol{x} \in \mathcal{G}} \left| f_{\hat{\boldsymbol{v}}_{S,i}}(\boldsymbol{x}) - f_{\boldsymbol{v}_{T,i}}(\boldsymbol{x}) \right|^2 , \tag{28}$$

$$\mathcal{L}_{\mathrm{field}}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x} \in \mathcal{P}_i} \left| f_{\hat{\boldsymbol{v}}_{S,i}}(\boldsymbol{x}) - f_{\boldsymbol{v}_{T,i}}(\boldsymbol{x}) \right|^2 . \tag{29}$$

The scalar outputs represent pixel intensities for MNIST and distance-field values for ModelNet40. Table 9 gives the training hyperparameters and number of query points.

Table 9: Operator training.

	ViT	CNN	MLP	MNIST INR	ModelNet40 INR
Objective	Eq. 27	Eq. 27	Eq. 27	Eq. 28	Eq. 29
Optimizer / decay	AdamW / 10^{-3}	AdamW / 10^{-3}	AdamW / 10^{-3}	AdamW / 10^{-3}	AdamW / 10^{-3}
Learning rate	$3{\cdot}10^{-4}$ or 10^{-3}	10^{-4}	10^{-4} / 10^{-3}	$5{\cdot}10^{-4}$–$2{\cdot}10^{-3}$	3–$5{\cdot}10^{-4}$
Warmup (steps)	none	200	200	none	none
Functional inputs	128 images	128 images	128 images	784 coordinates	4096 or 2048 points

All runs use global gradient clipping at 1.

D.2 Evaluation protocol

Teacher, anchor, and predicted student. We follow the notation of Section 5.1. For each held-out teacher $\boldsymbol{v}_T$, we construct the anchor $\boldsymbol{v}_0$ using the corresponding method in Table 10 and predict $\hat{\boldsymbol{v}}_S = \mathrm{CrossGMN}_\theta(\boldsymbol{v}_T, \boldsymbol{v}_0)$. Let $\boldsymbol{v}_S^a(k)$ denote the student parameters after k distillation steps, where $a \in \{\mathrm{anchor}, \mathrm{CrossGMN}\}$ identifies the initialization:

$$\boldsymbol{v}_S^{\mathrm{anchor}}(0) = \boldsymbol{v}_0, \qquad \boldsymbol{v}_S^{\mathrm{CrossGMN}}(0) = \hat{\boldsymbol{v}}_S.$$

Both students are distilled toward the same teacher with the same optimizer, minibatches, data order, and evaluation steps. The comparison therefore measures the benefit of refining the anchor with CrossGMN. Additional structured-pruning baselines and random initialization appear in the reach–budget plots.

Table 10: Anchor initializations used for the distillation comparisons.

Family	Reduction axis	Anchor
MNIST SIREN	width / depth / width+depth	magnitude pruning / layer fold / fold, then prune
ModelNet40 SIREN	width / depth / width+depth	magnitude pruning / strided layer transfer / strided prune
CIFAR-10 MLP	width / depth or width+depth	magnitude pruning / layer fold (scale 0.1)
Heterogeneous MLPs	all	layer fold (scale 0.1)
CIFAR-10 ViT	attention dimension / head count / depth / head count+depth	top-k attention dimensions / head drop / strided block transfer / strided blocks + head drop
CNN	channel width	magnitude pruning

Student initializations. We define the structured initializations used as anchors and comparison baselines below; Table 10 identifies the anchor for each setting. Write the teacher's affine layers as $(\boldsymbol{W}_\ell, \boldsymbol{b}_\ell)_{\ell=1}^{L_T}$, with hidden widths d_ℓ, and let L_S be the student's number of affine layers. Retained unit indices are kept in increasing order.

Width reduction. When $L_S = L_T$, selecting m_ℓ units in each hidden layer gives index sets $S_\ell \subseteq \{1, \dots, d_\ell\}$ with $|S_\ell| = m_\ell$. With S_0 and S_{L_T} containing all input and output indices, the student parameters are

$$\boldsymbol{W}_\ell^0 = \boldsymbol{W}_\ell[S_\ell, S_{\ell-1}], \qquad \boldsymbol{b}_\ell^0 = \boldsymbol{b}_\ell[S_\ell].$$

Magnitude-based pruning (Xu et al., 2024; Li et al., 2017) and activation-based pruning (Hu et al., 2016) motivate the following variants. For each hidden layer, our implementations keep the m_ℓ units with the largest scores:

$$s_{\ell i}^{\text{magnitude}} = \textstyle\sum_j |(\boldsymbol{W}_\ell)_{ij}| + \sum_j |(\boldsymbol{W}_{\ell+1})_{ji}|, \tag{30}$$

$$s_{\ell i}^{\ell_1\text{-outgoing}} = \textstyle\sum_j |(\boldsymbol{W}_{\ell+1})_{ji}|, \tag{31}$$

$$s_{\ell i}^{\text{magnitude-sum}} = \textstyle\sum_j |(\boldsymbol{W}_\ell)_{ij}|, \tag{32}$$

$$s_{\ell i}^{\text{activation}} = \frac{1}{n_{\text{cal}}} \sum_{r=1}^{n_{\text{cal}}} \left| [\sigma(\boldsymbol{W}_\ell \boldsymbol{x}_{\ell-1,r} + \boldsymbol{b}_\ell)]_i \right|. \tag{33}$$

The magnitude anchor uses a two-sided ℓ_1 score, summing incoming and outgoing weight magnitudes. Here $\boldsymbol{x}_{\ell-1,r}$ is the input to layer ℓ for calibration example r, and σ is its activation. Two baselines use no score: *uniform* selection retains

$$S_\ell = \left\{ 1 + \text{round}\left(\frac{r(d_\ell - 1)}{m_\ell - 1} \right) : r = 0, \dots, m_\ell - 1 \right\}$$

for $m_\ell > 1$ (and $S_\ell = \{1\}$ for $m_\ell = 1$); *random-neuron* selection samples m_ℓ distinct indices using a fixed seed per teacher.

CNN magnitude pruning. For the CNN kernels K_ℓ of Appendix B.2.3, the anchor uses the convolutional analogue of the MLP magnitude score. For output channel c of layer ℓ, followed by another convolution, this is

$$s_{\ell c}^{\text{magnitude}} = \sum_{i,r,t} |(K_\ell)_{cirt}| + \sum_{o,r,t} |(K_{\ell+1})_{ocrt}|,$$

where i, o index channels and r, t index spatial kernel entries. For the final convolution, the outgoing term uses the first dense layer's weights associated with that channel. We retain the highest-scoring channels in increasing index order and slice biases and adjacent weights consistently. Thus the anchor uses structured magnitude pruning with a two-sided ℓ_1 score; the incoming term alone is the filter criterion of Li et al. (2017). For ViTs, structured selection acts on heads or attention coordinates and slices the corresponding projection blocks.

Structured OBS/OBC. This baseline removes units sequentially while correcting the surviving weights using a quadratic layer-reconstruction objective (Hassibi & Stork, 1992; Frantar & Alistarh, 2022). For hidden layer ℓ, let $\boldsymbol{A} \in \mathbb{R}^{n_{\text{cal}} \times d_\ell}$ contain its calibration activations as rows, and define

$$\boldsymbol{H}_0 = \frac{\boldsymbol{A}^\top \boldsymbol{A}}{n_{\text{cal}}}, \qquad \bar{h} = \frac{\text{tr}(\boldsymbol{H}_0)}{d_\ell}, \qquad \boldsymbol{H} = \boldsymbol{H}_0 + 10^{-2} \bar{h} \boldsymbol{I}.$$

Starting from $\boldsymbol{W} = \boldsymbol{W}_{\ell+1}$ and $\boldsymbol{K} = \boldsymbol{H}^{-1}$, each iteration selects and compensates for a removed unit by

$$p = \arg\min_i \frac{\|\boldsymbol{W}[:, i]\|_2^2}{\boldsymbol{K}_{ii}}, \qquad \boldsymbol{W} \leftarrow \boldsymbol{W} - \frac{\boldsymbol{W}[:, p] \boldsymbol{K}[p, :]}{\boldsymbol{K}_{pp}}. \tag{34}$$

Delete column p of $\boldsymbol{W}$ and the corresponding row and bias entry in the preceding affine map, and update the inverse on the remaining indices:

$$\boldsymbol{K} \leftarrow \left(\boldsymbol{K} - \frac{\boldsymbol{K}[:, p] \boldsymbol{K}[p, :]}{\boldsymbol{K}_{pp}} \right)_{-p, -p}.$$

Repeat until m_ℓ units remain.

Depth and joint reduction. For $2 \le L_S < L_T$, *strided layer transfer* copies student layer j from teacher layer

$$\pi(j) = 1 + \mathrm{round}\left(\frac{(j-1)(L_T-1)}{L_S-1}\right), \qquad j = 1, \dots, L_S. \tag{35}$$

Layer fold instead replaces each contiguous group $\ell_1, \dots, \ell_2$ by the affine composition obtained by omitting its intermediate nonlinearities:

$$\boldsymbol{W}^0 = c\, \boldsymbol{W}_{\ell_2} \cdots \boldsymbol{W}_{\ell_1}, \qquad \boldsymbol{b}^0 = c\left(\boldsymbol{b}_{\ell_2} + \sum_{r=\ell_1}^{\ell_2-1} \boldsymbol{W}_{\ell_2} \cdots \boldsymbol{W}_{r+1} \boldsymbol{b}_r\right). \tag{36}$$

At $c = 1$ these are exactly the parameters of that affine composition; the scale controls its magnitude. Folded classifier maps use $c = 0.1$, whereas INRs use $c = 1$. *Keep-first* copies the first $L_S - 1$ teacher layers and initializes a new output layer; *keep-last* copies the last $L_S - 1$ layers and initializes a new input layer; *keep-first+last* retains layers at both ends and drops intermediate ones. For width+depth reduction, *fold-then-prune* applies magnitude pruning after folding, and *strided-prune* applies it after strided layer selection. Fresh random initialization uses the corresponding model family's initialization rule. Family-specific choices are given below.

Quality and target. Write $q^a_{\boldsymbol{v}_T}(k)$ for the quality of $\boldsymbol{v}^a_S(k)$, with larger values indicating better agreement with the teacher. For classifiers, this is the fraction of test inputs on which teacher and student predict the same label, measured on the full test set for ViTs and CNNs and a fixed 1,000-image subset for MLPs. For INRs, it is the negative MSE between student and teacher renders or fields. Both initializations are evaluated on the step grid $\mathcal{G} = \{0, 10, 20, \dots\}$, without interpolation.

For each teacher, we determine the anchor's plateau step $p_{\boldsymbol{v}_T}$ using the improvement threshold $\epsilon = 10^{-4}$ and patience $P = 50$ evaluations from Section 5.1. Its target quality is

$$q^\star_{\boldsymbol{v}_T} = \max_{\substack{k \in \mathcal{G} \\ k \le p_{\boldsymbol{v}_T}}} q^{\mathrm{anchor}}_{\boldsymbol{v}_T}(k).$$

Thus each predicted student is evaluated against the quality attained by its own teacher's anchor.

Budget and per-teacher speedup. Let $\mathcal{T}$ be the set of N held-out teachers in a compression setting. The setting's budget is the empirical 95th percentile of the anchor plateau steps, rounded up to the next evaluation step:

$$B = 10 \left\lceil \frac{Q_{0.95}(\{p_{\boldsymbol{v}_T} : \boldsymbol{v}_T \in \mathcal{T}\})}{10} \right\rceil. \tag{37}$$

The CrossGMN distillation budget is $H = 1.5B$. For $a \in \{\mathrm{anchor}, \mathrm{CrossGMN}\}$, define the first recorded step reaching the target by

$$k^a_{\boldsymbol{v}_T} = \min\left\{k \in \mathcal{G} : k \le h^a_{\boldsymbol{v}_T},\ q^a_{\boldsymbol{v}_T}(k) \ge q^\star_{\boldsymbol{v}_T}\right\}, \qquad h^{\mathrm{anchor}}_{\boldsymbol{v}_T} = p_{\boldsymbol{v}_T}, \quad h^{\mathrm{CrossGMN}}_{\boldsymbol{v}_T} = H,$$

where the minimum is over recorded evaluations and $\min \varnothing = +\infty$. The anchor reaches its target by construction. Following the main paper, the per-teacher speedup is

$$S_{\boldsymbol{v}_T} = \frac{k^{\mathrm{anchor}}_{\boldsymbol{v}_T}}{k^{\mathrm{CrossGMN}}_{\boldsymbol{v}_T}}.$$

If no recorded CrossGMN evaluation reaches the target by H, we count a failure and set $S_{\boldsymbol{v}_T} = 0$.

Reported metrics. We report the median and interquartile range of $\{S_{\boldsymbol{v}_T} : \boldsymbol{v}_T \in \mathcal{T}\}$, together with the acceleration and failure rates defined in the main paper:

$$r_{\mathrm{acc}} = \frac{1}{N} \sum_{\boldsymbol{v}_T \in \mathcal{T}} \mathbf{1}[S_{\boldsymbol{v}_T} > 1], \qquad r_{\mathrm{fail}} = \frac{1}{N} \sum_{\boldsymbol{v}_T \in \mathcal{T}} \mathbf{1}[S_{\boldsymbol{v}_T} = 0].$$

All N teachers, including failures, contribute to these statistics; tables express the rates as percentages. The additional $> 2\times$ column in Table 17 uses the same definition with threshold 2. The reach–budget plots instead track fidelity to the teacher and performance retention against teacher-relative thresholds, as described in Section 5.1 and Appendix D.9.

Wall-clock speedup. We include the time needed to construct each student initialization. Let c_{anchor} be the anchor-construction time, c_{CrossGMN} the complete CrossGMN initialization time (including anchor construction, graph construction, and the forward pass), and $d_{\mathrm{anchor}}, d_{\mathrm{CrossGMN}}$ the measured distillation time per step. The per-teacher wall-clock speedup is

$$S^{\mathrm{wall}}_{\boldsymbol{v}_T} = \frac{c_{\mathrm{anchor}} + d_{\mathrm{anchor}} k^{\mathrm{anchor}}_{\boldsymbol{v}_T}}{c_{\mathrm{CrossGMN}} + d_{\mathrm{CrossGMN}} k^{\mathrm{CrossGMN}}_{\boldsymbol{v}_T}}. \tag{38}$$

Table 17 reports the median over all teachers, assigning wall-clock speedup 0 to failures. All timing constants were measured with device synchronization and batch size one on NVIDIA A100-SXM4-40GB (A100-SXM4-80GB for ViT). The ModelNet40 width+depth setting has no recorded runtime profile.

Table 11: Evaluation-time distillation. Classifiers use pure KL at temperature 1 and no learning-rate schedule; all evaluation loops record every 10 steps.

	ViT	CNN	MLP	MNIST INR	ModelNet40 INR
Optimizer	AdamW	AdamW	AdamW	Adam	Adam
Learning rate	10^{-3}	10^{-3}	10^{-3}	10^{-2}	10^{-2}
Batch	256 images	256 images	256 images	full 28×28 grid	4096 points
Augmentation	none	none	crop + flip	–	–
Cohort	1,000	400	1,000 (500 OOD)	1,000	1,000

D.3 MNIST AND FASHIONMNIST INR COMPRESSION

Population and splits. We use the pre-fit MNIST SIREN population of Navon et al. (2023): one INR per 28×28 image, with teacher architecture $[2, 32, 32, 1]$ (1,185 parameters), two sine hidden layers with $\omega_0 = 30$, and a linear output. Of 70,000 INRs, 55,000 are used for operator training, 5,000 for validation, and a fixed 1,000-INR subset of the test split for every reported evaluation. The FashionMNIST transfer population has the same architecture and $55{,}000/5{,}000/10{,}000$ split. The MNIST-trained operators are applied to its 1,000-INR test cohort without retraining.

Table 12: MNIST SIREN configurations. Compression is the teacher-to-student parameter-count ratio; s is the CrossGMN residual scale.

Reduction axis	Student	Params	Compression	Anchor	Operator lr	s
Width	$[2, 16, 16, 1]$	337	$3.52\times$	magnitude pruning	10^{-3}	.005
Width	$[2, 8, 8, 1]$	105	$11.29\times$	magnitude pruning	$2{\cdot}10^{-3}$	.005
Width	$[2, 4, 4, 1]$	37	$32.03\times$	magnitude pruning	10^{-3}	.005
Depth	$[2, 32, 1]$	129	$9.19\times$	layer fold	$5{\cdot}10^{-4}$	.01
Width+depth	$[2, 16, 1]$	65	$18.23\times$	fold, then prune	$5{\cdot}10^{-4}$	.01

Anchors and comparator initializations. Table 12 specifies the anchors; their formulas and the comparison baselines are defined in Appendix D.2. For MNIST depth reduction, layer folding retains the first affine map and composes the final two, omitting the intermediate sine activation. The reach–budget plots compare CrossGMN with these anchors, the width or depth baselines appropriate to each setting, and fresh SIREN initialization, all distilled with the same optimizer and minibatches. Width reductions use strided cross edges; depth and width+depth reductions use all-to-all cross edges.

Operator training and evaluation. The five operators use the MNIST row of Tables 8 and 9. One epoch contains 1,719 optimizer steps over the 55,000 training teachers; validation render MSE selects the checkpoint. The selected checkpoints are from epochs $29, 142, 30, 12$, and 40 in the order of Table 12. Evaluation distills the complete 1,000-teacher cohort as a batch of independent student parameter vectors and records per-INR render MSE every ten steps.

D.4 MODELNET40 3-D INR COMPRESSION

Population and splits. We use the pre-fit SIREN distance-field population of Shamsian et al. (2024), with teacher architecture $[3, 48, 48, 48, 48, 1]$ (7,297 parameters) and $\omega_0 = 30$. The dataset

contains 12,311 objects, each fit ten times; we use view 0 and the object-disjoint $7{,}874/1{,}969/2{,}468$ train/validation/test split.

Table 13: ModelNet40 configurations. Columns follow Table 12.

Reduction axis	Student	Params	Compression	Anchor	Operator lr	s
Width	$[3, 24, 24, 24, 24, 1]$	1,921	$3.80\times$	magnitude pruning	$3{\cdot}10^{-4}$	.005
Depth	$[3, 48, 48, 1]$	2,593	$2.81\times$	strided layer transfer	$3{\cdot}10^{-4}$	.02
Depth	$[3, 48, 1]$	241	$30.28\times$	strided layer transfer	$5{\cdot}10^{-4}$	.02
Width+depth	$[3, 24, 24, 1]$	721	$10.12\times$	strided prune	$3{\cdot}10^{-4}$	.01

Anchors, training, and evaluation. We use magnitude pruning for width reduction, strided layer transfer for depth reduction, and strided-prune for width+depth reduction, as defined in Appendix D.2. Three operators use the sign-equivariant $d_{\text{hid}} = 192$ backbone. The width+depth operator uses the parameter-matched permutation-equivariant $d_{\text{hid}} = 324$ backbone. Each is trained for 30 epochs with a fixed set of uniformly sampled points and the final epoch is evaluated.

All settings are evaluated on $4{,}096$ points.

D.5 CIFAR-10 MLP COMPRESSION

Teacher populations. For fixed-architecture compression, we construct a model zoo of $6{,}100$ ReLU MLP teachers of architecture $[3072, 64, 64, 64, 64, 10]$ (209,802 parameters), trained for 100 epochs with varied learning rates, weight decay, label smoothing, dropout, and seeds. The split contains $5{,}000$ training, 100 validation, and $1{,}000$ test teachers.

For heterogeneous-architecture compression, we construct a second ReLU MLP model zoo with four training architectures: $[3072, 16, 10]$, $[3072, 16, 16, 10]$, $[3072, 32, 10]$, and $[3072, 32, 32, 10]$, all mapped to the common student $[3072, 8, 10]$ as in Section 5.2. We hold out the wider architecture $[3072, 64, 10]$ and the deeper architecture $[3072, 16, 16, 16, 16, 10]$ to test generalization to unseen source architectures. The respective splits contain $5{,}000$ training, 100 validation, $1{,}000$ test teachers from the training architectures, and 500 teachers from unseen architectures, balanced across the architectures in each split. These model zoos and their fixed splits can also serve as benchmarks for other weight-space methods, supporting evaluation within an architecture and generalization to unseen architectures.

Table 14: MLP compression configurations.

Reduction axis	Student	Params	Compression	Anchor	Operator lr	s
Depth	$[3072, 64, 64, 10]$	201,482	$1.04\times$	layer fold (scale 0.1)	10^{-4}	.1
Width	$[3072, 32, 32, 32, 32, 10]$	101,834	$2.06\times$	magnitude pruning	10^{-4}	.1
Width+depth	$[3072, 32, 32, 10]$	99,722	$2.10\times$	layer fold (scale 0.1)	10^{-4}	.1
Heterogeneous architectures	$[3072, 8, 10]$	24,674	2.00–$4.04\times$	layer fold (scale 0.1)	10^{-3}	.1

Anchors. We use the magnitude-pruning and layer-fold initializations defined in Appendix D.2, with the choices in Table 14. Layer folding partitions the teacher's affine layers into contiguous groups, composes each group, and scales folded interior maps by 0.1.

Operator training and evaluation. The fixed-architecture operators use eight message-passing layers and learning rate 10^{-4}. The heterogeneous-architecture operator uses four layers and learning rate 10^{-3}. The test cohorts contain $1{,}000$ teachers from training architectures or 500 teachers from unseen architectures.

D.6 CIFAR-10 VISION TRANSFORMER COMPRESSION

Teacher population. We construct a model zoo of $6{,}100$ ViTs with patch size 4, residual width 48, four blocks, two heads of dimension 24, FFN width 192, and a ten-class head (115,930 parameters). Teachers vary in optimizer hyperparameters and seed. The splits contain $5{,}000$ training, 100 validation, and $1{,}000$ test teachers. Like the MLP zoos, this population and its fixed splits can serve as a benchmark for learning from model parameters, extending evaluation to the Transformer family. Operator configurations and checkpoints are chosen using only the validation split.

Table 15: ViT compression configurations. All four settings use the strided cross-edge policy of Definition 5; when the teacher and student block counts agree it reduces to shared-index matching. Here d is the residual width, L the number of blocks, H the number of heads, and d_a the total attention dimension across heads.

Reduction axis	Student	Params	Compression	Anchor	Operator lr	s
Attention dimension	$d = 48, L = 4, H = 2, d_a = 24$	97,210	$1.19\times$	top-k attention dimensions	$3{\cdot}10^{-4}$	.1
Head count	$d = 48, L = 4, H = 1, d_a = 24$	97,210	$1.19\times$	head drop	10^{-3}	.1
Depth	$d = 48, L = 2, H = 2, d_a = 48$	59,386	$1.95\times$	strided block transfer	$3{\cdot}10^{-4}$	.1
Head count+depth	$d = 48, L = 2, H = 1, d_a = 24$	50,026	$2.32\times$	strided blocks + head drop	10^{-3}	.1

Anchors, training, and evaluation. Attention-dimension and head-count anchors use two-sided ℓ_1 magnitude pruning. Each attention coordinate is scored by summing the absolute weights in its value-projection row and output-projection column. Attention-dimension reduction retains the twelve highest-scoring coordinates in each head; head-count reduction retains the head with the largest sum of coordinate scores. The retained indices are applied to the query, key, and value rows and output columns, preserving their original order. Depth reduction uses strided block selection, retaining the first and third teacher blocks; head count+depth reduction first selects these blocks and then prunes their heads. Parameters outside the reduced modules are copied unchanged.

Each operator is trained for $10{,}000$ steps at three candidate learning rates; mean validation agreement selects the run and checkpoint. The selected checkpoints are from steps $10{,}000$, $9{,}500$, $9{,}000$, and $10{,}000$, in the order of Table 15.

D.7 MNIST CNN COMPRESSION

The public zoo of Schürholt et al. (2022b) contains $2{,}560$ classifiers with three convolutional stages of widths $[8, 6, 4]$, kernel sizes $5, 5, 2$, pooling after the first two stages, a 20-unit fully connected layer, and a ten-class head ($2{,}464$ parameters). The population spans tanh, ReLU, sigmoid, and GELU models; the activation-type embedding is therefore enabled.

We reduce only convolutional width, $[8, 6, 4] \to [4, 3, 2]$, leaving the head unchanged and retaining 41.5% of the parameters. The anchor uses the two-sided ℓ_1 magnitude-pruning criterion in Appendix D.2. The operator uses the CNN row of Tables 8 and 9, with residual scale $s = .4$, and is trained for $100{,}000$ steps on $1{,}407$ teachers. Evaluation uses a fixed 400-teacher cohort.

D.8 ABLATIONS

All rows in Table 16 use the MNIST INR reduction $[2, 32, 32, 1] \to [2, 8, 8, 1]$ and the same $1{,}000$ held-out teachers as the corresponding main result. Each row changes one choice from the reference CrossGMN configuration and follows the per-teacher speedup protocol in Appendix D.2.

The communication rows isolate the role of the joint graph: direct neuron-level cross edges outperform removing communication or replacing it by a pooled FiLM summary, while strided and all-to-all cross-edge policies yield similar speedups. The remaining rows show that performance depends more strongly on the training population, message-passing depth, and width than on choosing ScaleGMN rather than a parameter-matched GMN. Direct teacher matching is also more effective here than supervising the initialization toward a student obtained after a fixed number of distillation steps. These are single operator runs; the table should not be read as an estimate of seed variance.

D.9 REACH–BUDGET PLOTS

The speedup tables use the target $q^\star_{\boldsymbol{v}_T}$ attained by each teacher's anchor. To show progress independently of that relative target, the following figures plot the fraction of teachers that have reached a fixed teacher-dependent criterion by distillation budget b. They cover exactly the 22 settings reported in the main tables: all rows of Tables 1, 2 and 4, and the pooled rows of Tables 3 and 5.

For INRs, *fidelity reach* requires $\mathrm{PSNR}(\text{student}, \text{teacher}) \geq T$, while *performance-retention reach* requires $\mathrm{PSNR}(\text{student}, \text{data}) \geq \mathrm{PSNR}(\text{teacher}, \text{data}) - \varepsilon$. For classifiers, fidelity is test agreement at least T, and retention is student accuracy at least $(1 - \varepsilon)$ times that teacher's accuracy. For ModelNet40, we record only error to the teacher field and therefore report fidelity reach only.

Table 16: **CrossGMN ablations on MNIST INR compression.** Median speedup and IQR relative to the corresponding anchor, including failures ($S_{\boldsymbol{v}_T} = 0$).

Ablation	Variant	Speedup	$> 1\times$ (%)	Failed (%)
Communication	Strided	$5.20\times_{[1.1,12.5]}$	76	18
	No cross edges	$3.18\times_{[0.4,8.6]}$	67	24
	All-to-all	$5.13\times_{[0.9,11.6]}$	75	18
	FiLM on pooled teacher (Perez et al., 2018)	$3.22\times_{[0.4,9.1]}$	68	24
Initialization	Structured pruning	$5.20\times_{[1.1,12.5]}$	76	18
	Random initialization	$3.76\times_{[0.5,8.5]}$	70	23
Supervision	Teacher matching	$5.20\times_{[1.1,12.5]}$	76	18
	Student after 50 steps	$1.15\times_{[0.0,4.0]}$	52	35
	Student after 250 steps	$2.35\times_{[0.0,6.9]}$	61	30
	Student after 2,500 steps	$1.41\times_{[0.0,4.6]}$	55	33
Training teachers	55,000	$5.20\times_{[1.1,12.5]}$	76	18
	1,000	$1.67\times_{[0.0,5.1]}$	58	33
	500	$1.01\times_{[0.0,3.8]}$	50	36
	100	$0.00\times_{[0.0,2.1]}$	36	51
Backbone	ScaleGMN	$5.20\times_{[1.1,12.5]}$	76	18
	parameter-matched GMN	$5.70\times_{[1.0,13.0]}$	76	18
Capacity	width 128, 4 layers	$5.20\times_{[1.1,12.5]}$	76	18
	width 64, 4 layers	$1.83\times_{[0.0,5.8]}$	58	32
	width 32, 4 layers	$0.69\times_{[0.0,3.4]}$	44	43
	width 128, 3 layers	$1.58\times_{[0.0,5.8]}$	57	32
	width 128, 2 layers	$0.62\times_{[0.0,3.0]}$	43	44

We choose thresholds to make the reach–budget plots informative: overly easy or difficult thresholds can produce nearly flat curves near one or zero, respectively, obscuring how reach changes with the distillation budget. We choose a threshold separately for each setting and criterion from

$$T_{\mathrm{INR}} \in \{15, 20, 25, 30, 35, 40\}\ \mathrm{dB}, \quad \varepsilon_{\mathrm{INR}} \in \{.5, 1, 2, 3, 5, 7, 10, 15, 20\}\ \mathrm{dB},$$

$$T_{\mathrm{clf}} \in \{.95, .9, .85, .8, .75, .7, .6, .5, .4, .3\}, \quad \varepsilon_{\mathrm{clf}} \in \{.01, .02, .05, .1, .15, .2, .3, .4, .5\}.$$

For each setting, let H_{record} be the last recorded distillation step. We choose the strictest threshold for which (i) the anchor reaches it for at least half the teachers by H_{record}, (ii) neither the anchor nor CrossGMN reaches it for half the teachers at step 0, and (iii) the anchor's median step to reach it lies in $[H_{\mathrm{record}}/200, H_{\mathrm{record}}/3]$. If none qualifies, we display the most informative available level and mark it by an asterisk.

Each curve is an empirical step function giving the fraction of teachers whose students have reached the threshold by budget b. A final value below one indicates that some students never reach it within the recorded horizon. Each panel shows the budget interval in which its curves change.

We show all evaluated initializations: CrossGMN in solid blue, its anchor in dashed orange, random initialization in dotted gray, and additional structured baselines in thin gray. Appendix D.2 defines the baselines, and the family-specific subsections specify which ones are evaluated. In thirteen of the forty panels, both CrossGMN and its anchor eventually reach the threshold for nearly all teachers.

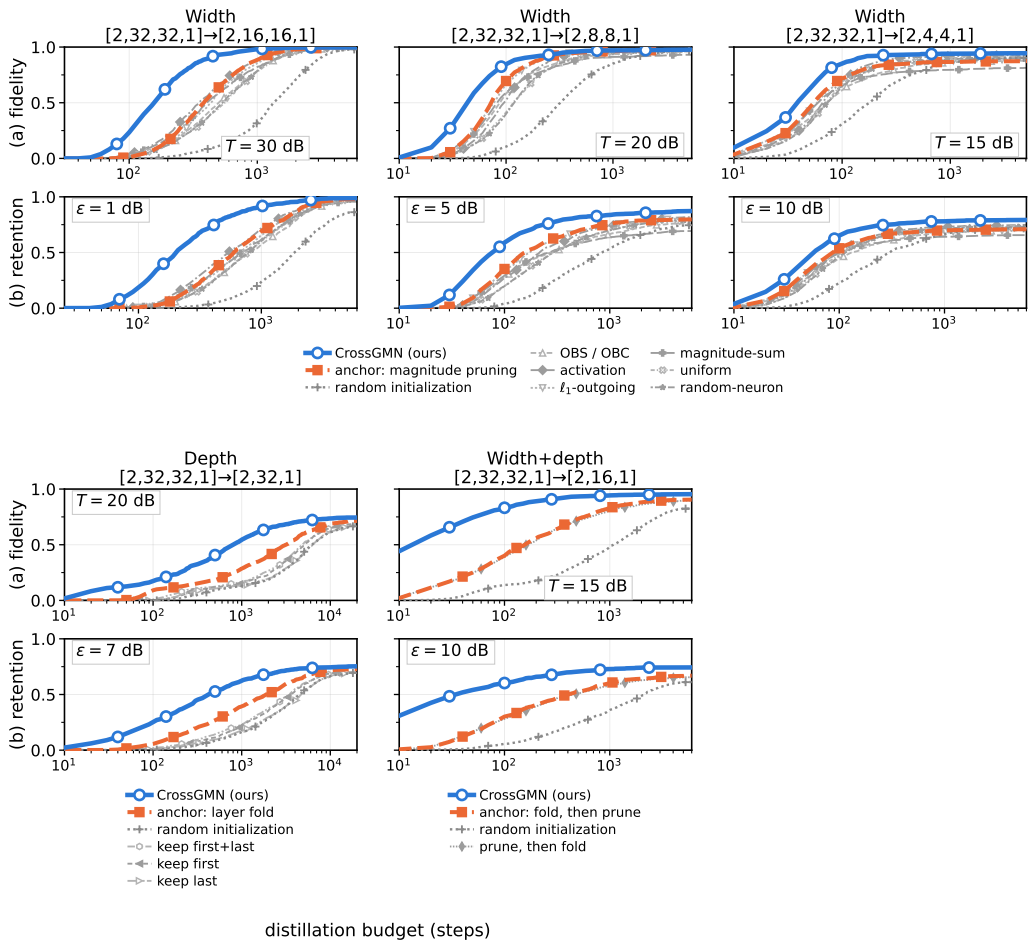


Figure 4: **MNIST INR reach–budget plots** ($N = 1{,}000$ per setting). All five reductions in Table 1; fidelity reach above and performance-retention reach below.

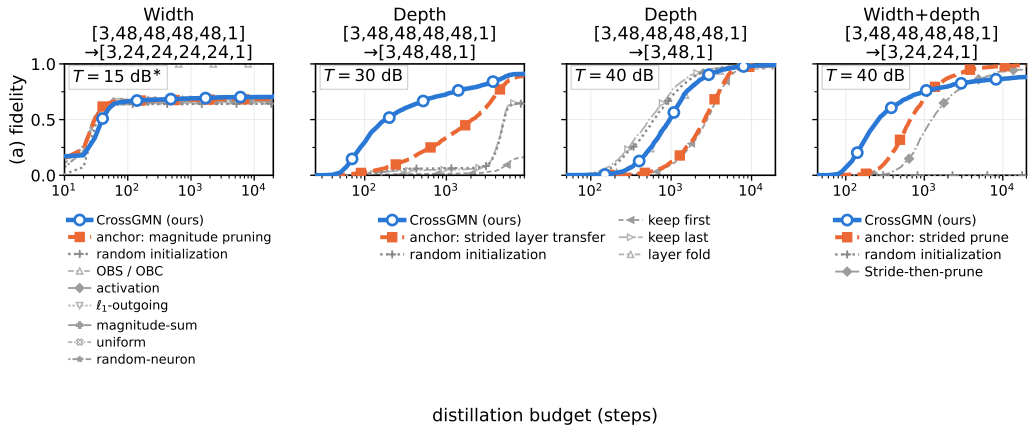


Figure 5: **ModelNet40 INR reach–budget plots** ($N = 1{,}000$ per setting). All four reductions in Table 1, showing fidelity to the teacher field. The asterisk marks a fallback threshold.

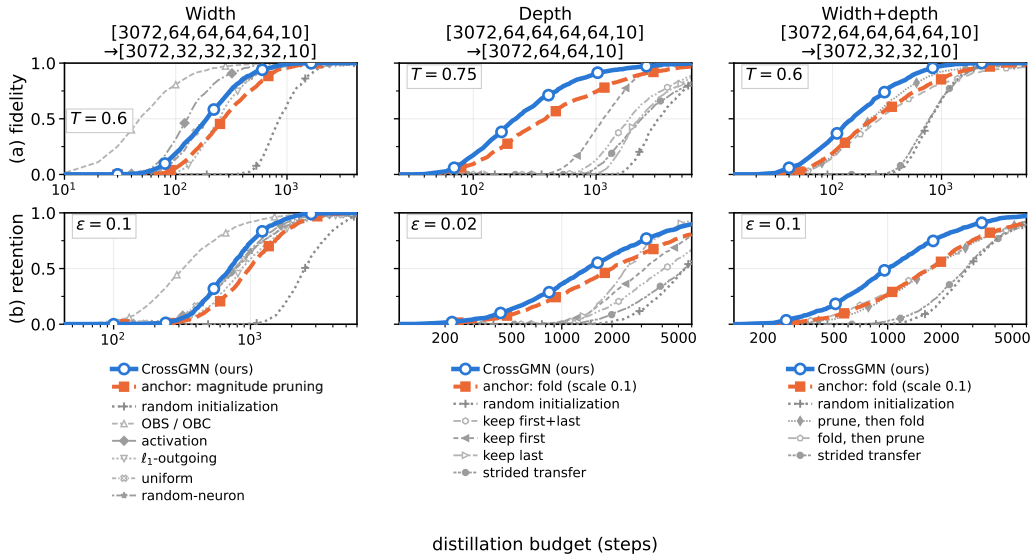


Figure 6: **CIFAR-10 MLP reach–budget plots** ($N = 1{,}000$ per setting). The three reductions in Table 2; fidelity reach above and performance-retention reach below.

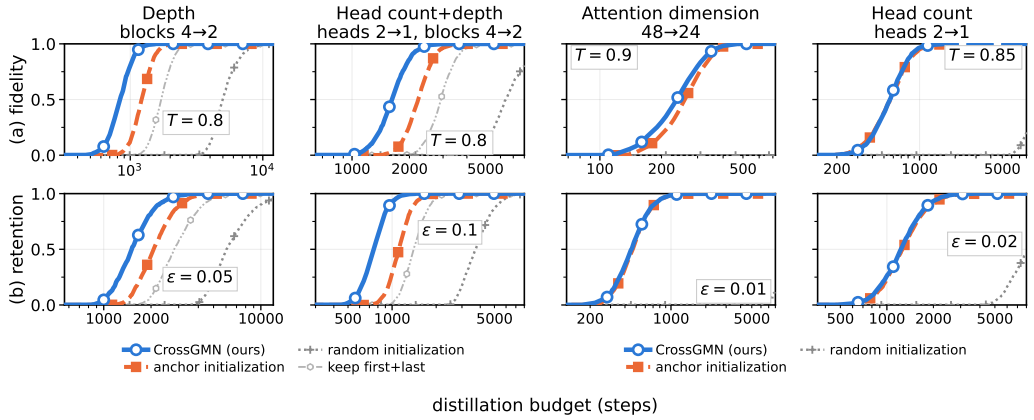


Figure 7: **CIFAR-10 ViT reach–budget plots** ($N = 1{,}000$ per setting). The four reductions in Table 2; fidelity reach above and performance-retention reach below.

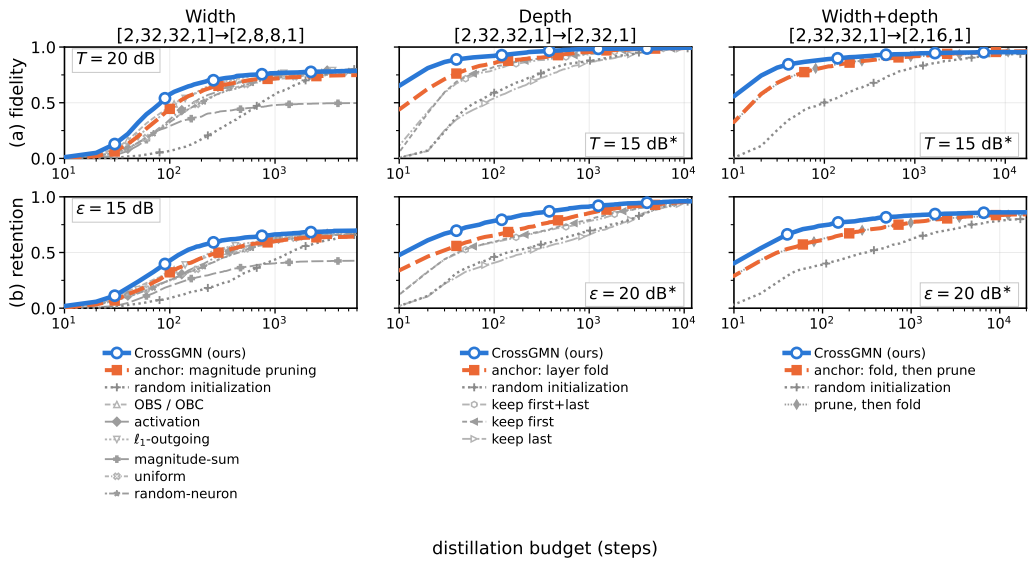
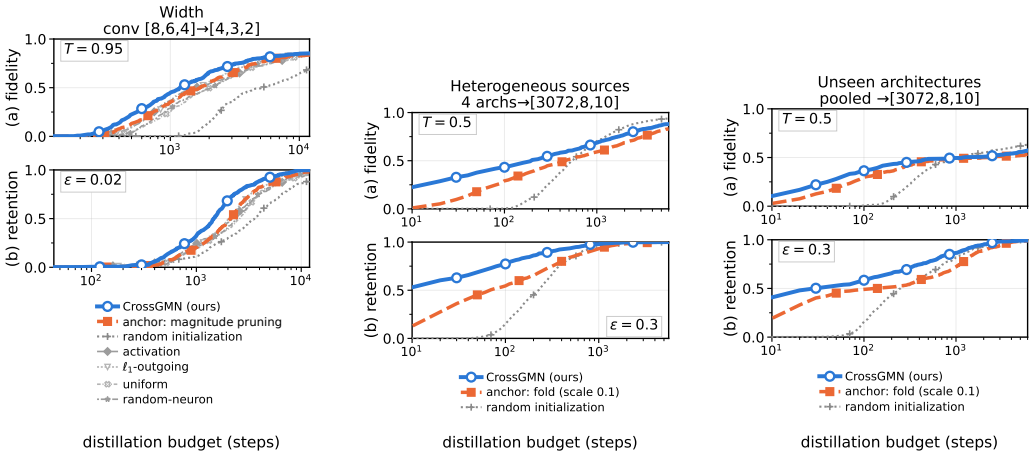


Figure 8: **Transfer across datasets** ($N = 1{,}000$ per setting). MNIST-trained CrossGMN operators applied to FashionMNIST INRs without retraining (Table 4). Asterisks mark fallback thresholds.



(a) MNIST CNN classifiers (b) heterogeneous MLPs (c) unseen architectures

Figure 9: **Reach–budget plots for CNNs and heterogeneous MLPs.** (a) CNN width reduction ($N = 400$, Table 2). (b) Four teacher architectures mapped to $[3072, 8, 10]$ ($N = 1{,}000$, Table 3). (c) The same operator on unseen source architectures ($N = 500$, Table 5).

D.10 COMPLETE RESULTS

Table 17 collects the complete results under the protocol in Appendix D.2. It reports each setting's anchor and budget, the median and interquartile range of per-teacher speedups, the fractions with $S_{\boldsymbol{v}_T} > 1$ and $S_{\boldsymbol{v}_T} > 2$, the failure rate, and wall-clock speedup including initialization. The final group is an anchor ablation.

Table 17: **Complete compression results.** Budget is B from Equation 37. Speedup statistics include failures ($S_{\boldsymbol{v}_T} = 0$). Wall-clock speedup is the median per-teacher ratio from Equation 38. Δ@Budget is the median paired CrossGMN-minus-anchor quality difference at the last evaluation step not exceeding B: PSNR (dB) for INRs and test agreement for classifiers.

Setting	Anchor	Budget	Median speedup	IQR	> 1 (%)	> 2 (%)	Failed (%)	Wall-clock speedup	Δ@ Budget
MNIST SIREN INRs									
width 3.5×	magnitude pruning	2,710	**2.20**×	[1.24, 3.86]	83	54	2	2.17×	+1.78
width 11.3×	magnitude pruning	2,480	**5.20**×	[1.14, 12.48]	76	68	18	4.99×	+1.65
width 32×	magnitude pruning	3,520	**4.10**×	[0.00, 14.10]	62	58	35	3.98×	+0.47
depth 2→1	layer fold	4,560	**4.07**×	[2.21, 6.57]	90	78	7	4.00×	+0.65
width+depth 18.2×	fold, then prune	2,750	**8.89**×	[3.56, 16.83]	86	83	13	8.36×	+0.44
FashionMNIST SIREN INRs (transfer across datasets)									
width 3.5×	magnitude pruning	2,740	**2.06**×	[1.15, 3.16]	80	51	6	2.04×	+0.77
width 11.3×	magnitude pruning	2,250	**2.18**×	[0.00, 5.77]	61	52	33	2.13×	+0.32
depth 2→1	layer fold	4,320	**3.09**×	[1.54, 6.09]	86	66	6	3.03×	+0.32
width+depth 18.2×	fold, then prune	2,870	**3.78**×	[1.11, 10.27]	76	64	17	3.68×	+0.17
ModelNet40 3-D INRs									
width 3.8× ($s = 0.005$)	magnitude pruning	1,750	**1.11**×	[0.88, 1.73]	57	20	18	0.98×	+0.00
depth 4→2	strided layer transfer	4,730	**6.85**×	[0.95, 19.81]	74	68	12	6.09×	+3.87
depth 4→1	strided layer transfer	1,930	**2.88**×	[1.97, 4.03]	96	74	0	2.70×	+2.61
width+depth 10.1×	strided prune	1,830	**2.46**×	[0.41, 4.76]	68	55	23	–	+0.29
MLP (CIFAR-10)									
width 64→32	magnitude pruning	3,940	**1.04**×	[0.75, 1.35]	54	4	5	1.01×	+0.0050
depth 4→2	layer fold (scale 0.1)	3,290	**1.25**×	[0.73, 2.27]	60	30	3	1.15×	+0.0120
width+depth	layer fold (scale 0.1)	3,780	**1.18**×	[0.76, 1.84]	59	21	3	1.11×	+0.0100
MNIST CNN									
conv width	magnitude pruning	9,900	**1.17**×	[0.93, 1.42]	68	7	1	1.16×	+0.0008
ViT (CIFAR-10)									
attention dimension 48→24	top-k attention dimensions	3,680	**0.74**×	[0.63, 0.85]	7	0	5	0.73×	−0.0011
head count 2→1	head drop	4,530	**0.86**×	[0.78, 0.96]	16	0	0	0.85×	+0.0004
depth 4→2	strided block transfer	6,960	**1.21**×	[1.12, 1.30]	95	0	0	1.20×	+0.0044
head count+depth	strided blocks + head drop	7,470	**1.18**×	[1.09, 1.27]	92	0	0	1.16×	+0.0047
Heterogeneous-architecture compression									
pooled (four source architectures)	layer fold (scale 0.1)	3,640	**1.28**×	[0.71, 2.25]	62	28	10	1.20×	+0.0040
Generalization to unseen source architectures									
pooled (unseen source architectures)	layer fold (scale 0.1)	3,450	**1.06**×	[0.49, 1.84]	51	23	14	1.00×	+0.0030
Anchor ablation: heterogeneous-architecture compression (layer-fold reference above)									
strided layer transfer	strided layer transfer	4,240	**1.34**×	[0.62, 2.60]	61	35	13	1.30×	+0.0030
keep first+last	keep first+last	4,190	**1.23**×	[0.54, 2.36]	60	31	15	1.20×	+0.0030
random initialization	random initialization	3,730	**0.82**×	[0.34, 1.59]	42	16	21	0.80×	+0.0020

The quality difference at B complements speedup: faster distillation need not imply a better asymptotic solution. All reported settings with median speedup above one also have nonnegative median quality differences at this budget. For ViT attention-dimension and head-count reductions, the struc-

tured anchors already provide strong initializations, and CrossGMN does not accelerate subsequent distillation.

D.11 Classifier teacher-population diversity

As noted in Section 5.1, amortizing a learned compression operator across a model population is motivated by the population spanning diverse functions. We therefore measure diversity in classifier teachers' outputs, which directly characterize the functions that their students must preserve. Parameter differences alone would be insufficient, since different weights can realize the same function.

Populations and sampling. Table 18 reports averages over 10,000 randomly sampled pairs of distinct teachers from each classifier population, using the full held-out cohort evaluated for compression. Settings that share a teacher population share its diversity statistics.

Diversity metrics. For teachers $\boldsymbol{v}_T$ and $\boldsymbol{v}'_T$, let $c_T(\boldsymbol{x}) = \arg\max_j f_{\boldsymbol{v}_T}(\boldsymbol{x})_j$ and $c_{T'}(\boldsymbol{x})$ be their predicted labels on a shared test set $\mathcal{D}$. *Prediction agreement* measures how often their decisions coincide:

$$p_o = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathbf{1}[c_T(\boldsymbol{x}) = c_{T'}(\boldsymbol{x})].$$

Lower agreement indicates greater variation in predictions. To distinguish shared correct predictions from shared mistakes, define each teacher's error set $E_T = \{(\boldsymbol{x}, y) \in \mathcal{D} : c_T(\boldsymbol{x}) \neq y\}$ and report *wrong-prediction intersection over union*:

$$\mathrm{IoU}_{\mathrm{err}}(T, T') = \frac{|E_T \cap E_{T'}|}{|E_T \cup E_{T'}|}.$$

This measures whether the teachers fail on the same inputs, regardless of which incorrect labels they predict. Lower values mean less overlap in errors, although the metric also depends on accuracy.

Finally, *Cohen's* κ (Cohen, 1960) corrects agreement for each teacher's predicted-label frequencies:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \qquad p_e = \sum_j p_T(j) p_{T'}(j), \qquad p_T(j) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathbf{1}[c_T(\boldsymbol{x}) = j].$$

Here p_e is the agreement expected from independent predictions with those label frequencies. Thus κ complements raw agreement by accounting for class preferences; it does not remove all effects of accuracy or task difficulty. Together, the three metrics describe overall prediction similarity, overlap in errors, and agreement beyond the label marginals.

Table 18: **Classifier teacher-population diversity.** Mean statistics over 10,000 pairs of distinct teachers. Lower agreement, error IoU, and κ indicate greater diversity in their respective aspects.

Teacher population	Teachers	Agreement ↓	Error IoU ↓	κ ↓
CIFAR-10 MLP	1,000	0.715	0.694	0.682
CIFAR-10 ViT	1,000	0.792	0.556	0.769
MNIST CNN	400	0.891	0.210	0.879
CIFAR-10 MLP, four source architectures	1,000	0.643	0.683	0.602
unseen source architectures	500	0.618	0.653	0.574

Findings. On CIFAR-10, fixed-architecture MLP teachers have lower agreement than ViTs (0.715 versus 0.792), also reflected in κ (0.682 versus 0.769). The MNIST CNNs have high agreement (0.891) but low error overlap (0.210): their predictions often coincide, while their mistakes are less shared. The heterogeneous MLP population and its unseen-architecture counterpart have similar statistics, with $\kappa = 0.602$ and 0.574, respectively. These results establish variation within each population.