TRANSFERABLE GRAPH METANETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

A *weight space network* (or *metanetwork*) takes the weights of another neural network as input and predicts properties of it. Most prior work trains such models on input networks of one or a few fixed sizes and evaluates them in-distribution. The few attempts at out-of-distribution size generalization remain limited in scope and have achieved only modest success. Consequently, the potential efficiency gains of training on small networks and evaluating on much larger ones remain largely unrealized. We propose *Transferable Graph Metanetworks*, which extend the graph metanetwork paradigm with a set of modifications that make performance transferable across input networks of different *widths*. The modifications follow two principles: invariance to the ways in which networks of different widths represent the same function, and continuity, such that weights representing similar functions receive similar predictions. We further study whether size generalization is possible for input networks trained independently from random initialization. Empirically, our modifications significantly improve size generalization on every task we consider. Performance is strongest on input networks trained under the maximal-update parameterization (μP), where it remains robust up to $42\times$ the training width. Theoretically, we explain these observations with infinite-width limit theory: we prove size-generalization guarantees for our model on μP-trained inputs, and explain why it can fail under other parameterizations.

1 INTRODUCTION

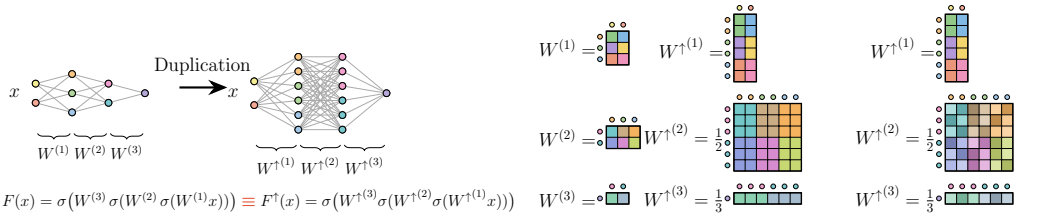
The parameters of a trained neural network are increasingly treated as a data modality in their own right (Han et al., 2026). Such data is plentiful and informative: public model repositories such as Hugging Face hold millions of checkpoints, and each one determines the function its network represents while carrying traces of its task, its learned representations, and the training process that produced it. What makes this data hard to work with is that it arrives as a long list of numbers in which no individual coordinate means anything on its own and in which many hidden symmetries relate different parameter settings to the same underlying function. As a result, the map from those numbers to the quantities one cares about is neither explicit nor simple. The approach that has emerged is to learn that map from data by feeding the parameters as input to a second neural network that exploits the underlying symmetries of the parameter space. This second network is trained either to predict the property of interest or to output new weights or weight updates. Such a model is called a *weight space network*, or *metanetwork*. This recipe supports a wide range of applications. For example, metanetworks can predict accuracy or the generalization gap from the weights without ever evaluating the network on data, making model selection and hyperparameter search cheaper (Unterthiner et al., 2020; Schürholt et al., 2021; Mehta et al., 2024). Another use case is the implicit neural representation (INR): each data point *is* a small network representing an image, a shape, or a scene, so that classifying or editing the data becomes an operation on parameters (Dupont et al., 2022; Luigi et al., 2023; Navon et al., 2023a).

We ask a question that prior work largely leaves open:

> Can we construct metanetworks that generalize across input networks of different sizes, and in particular that are trained on small networks and perform well on large ones?

For graph neural networks (GNNs), there is a vast literature on size-generalization Yehudai et al. (2021); Bevilacqua et al. (2021) and transferability Ruiz et al. (2020); Levie (2024); Maskey et al. (2023); Keriven et al. (2020); Velasco et al. (2024), but those results are not directly applicable

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.



(a) Duplicating hidden neurons leaves the function unchanged (b) Duplication widening (4) (c) Block-wise widening (6)

Figure 1: Function-preserving widenings of an MLP. (b) Copying and rescaling weights duplicates hidden neurons, as in (a), so the function is unchanged; duplication-compatible GMN is invariant to this transformation. (c) More generally, the copied weights can be redistributed across rows of the block, which still amounts to the duplication of neurons in (a) and preserves the function; matrix-product GMN is invariant to this transformation.

to the metanetwork setting due to the particular structure and symmetries of their inputs. Size generalization is useful in this context for two related reasons. First, real model populations are heterogeneous, spanning many widths and depths (Touvron et al., 2023), but checkpoints of the largest models are the rarest, precisely because they are the most expensive to train. Second, the cost of training a metanetwork grows with the size of its input networks. A metanetwork that trains on the small networks that are cheap and plentiful, and then applies to larger sizes it has never seen, avoids both obstacles.

Many existing metanetworks based on equivariant layers (Navon et al., 2023a;b; Zhou et al., 2023a;b; 2024; Tran et al., 2024) are designed specifically for a fixed weight space (on specific architecture with fixed width and depth). Graph metanetworks (GMNs) that construct graphs out of input networks and apply GNNs or attention (Lim et al., 2024; Kofinas et al., 2024; Kalogeropoulos et al., 2024; Bar-Shalom et al., 2026) do accept input networks of arbitrary size, and some are tested on networks a small factor wider than those seen in training, with only limited success reported (Lim et al., 2024; Kofinas et al., 2024). Indeed, size generalization was not a design goal for these models, and nothing in their architecture constrains their predictions at unseen sizes.

As our **first contribution** (Section 3), motivated by the theory of any-dimensional learning and transferability (Levin et al., 2026), we propose the framework of *Transferable Graph Metanetworks*: a set of modifications that make the performance of an existing GMN transferable across input networks of different *widths* (leaving depth to future work). The modifications come in two levels, and both are generic enough to apply to any standard GMN variants taking MLP and CNN input networks. All claimed properties are proved in Appendix B and verified numerically in Appendix D.1.

- **Level 1**. Networks of different widths can represent the same function, and a metanetwork should be invariant to the transformations that relate them. We begin with the simplest such transformation, copying and rescaling the weights accordingly (see Figure 1(b)), and show that three simple modifications already make a GMN exactly invariant to it: fan-in rescaling of the input weights, mean aggregation, and a layer-wise mean readout. We call the result a *duplication-compatible GMN*. This modification does not affect the expressivity at a fixed width.
- **Level 2**. The GMNs from Level 1 have two shortcomings. First, they are not invariant to the more general transformations that preserve the represented function (see Figure 1(c)). Second, they are not continuous, which may affect generalization behavior: a small perturbation of the input weights that barely changes the represented function can change the metanetwork's output substantially. We modify the GMNs to address both issues and call the result a *matrix-product GMN*. This modification might restrict expressivity, but we show that it improves size generalization.

Our **second contribution** (Section 4) is an empirical study of whether our proposed modifications improve size generalization in practical settings: can the GMNs generalize to input networks of a much larger width than the one it was trained on? In practice, the input networks of different widths are trained independently from random initializations, so how their training dynamics relate to each other across widths depends on both the *optimization* (the algorithm and its hyperparameters) and the *parameterization* (how the initialization and those optimization hyperparameters scale with width). We fix the optimizer and vary only the parameterization, considering two choices: the standard PyTorch default, called standard parameterization (SP), and maximal-update parameterization (μP) (Yang & Hu, 2021), a popular choice that was shown to align training dynamics across widths and to enable zero-shot hyperparameter transfer (Yang et al., 2021) (see Figure 8). Our experiments

show that (i) both Level 1 and Level 2 modifications significantly improve size generalization in all cases, under both parameterizations, (ii) size generalization is much better on μP-trained input networks than on SP-trained ones, and (iii) the combination of the Level 2 matrix-product GMN with μP-trained input performs well across the full range of test widths.

Our **third contribution** (Section 5) is a theoretical analysis that explains these empirical findings. Using results from the infinite-width limit literature (Littwin & Yang, 2023) to characterize the distribution of trained weights, we prove size generalization guarantees for the matrix-product GMN on μP-trained inputs, formalizing observation (iii), and identify the obstructions that explain why size generalization may otherwise fail. μP constrains no individual width; it is a rule for how the initialization and optimization hyperparameters change with width. So our guarantee places no restriction on the input networks used for training, but instead specifies the wider networks it provably generalizes to: those trained with the same optimizer under the hyperparameters this scaling prescribes. Some such specification is unavoidable, since if the wide networks' training dynamics bear no relation to the narrow ones' then nothing learned at the training width can transfer, whatever the architecture; μP makes the relation precise. Beyond this regime, we still expect our modifications to help,albeit with weaker performance and no theoretical guarantee, as the SP experiments indicate.

Our theory focuses on MLP input networks, but the modification recipe itself is architecture-agnostic, and our experiments apply it to both MLP and CNN input networks. We expect the theory to extend beyond MLPs and leave this to future work.

2 Preliminaries and related work

2.1 Weight-space symmetries, metanetworks, and graph metanetworks

Most architectures are overparameterized with many different weights computing the same function. Whenever the prediction target depends only on the *function* the input network expresses, the metanetwork should be invariant to the transformations that leave the represented function unchanged. State-of-the-art metanetworks are designed to exploit this. The simplest such transformation is **permutation symmetry**: permuting the corresponding rows and columns of two adjacent weight matrices permutes the hidden neurons between them and leaves the function unchanged. Another is **scale symmetry**: scaling a hidden neuron's incoming weights and bias while inversely scaling its outgoing weights leaves the function unchanged, with the admissible rescalings depending on the activation. Both symmetries act within a single weight space, at one fixed width.

There are three main approaches to designing metanetworks that respect these symmetries. The first builds equivariant layers with the standard toolkit of equivariant machine learning, as in DWS-Nets (Navon et al., 2023a), NFNs (Zhou et al., 2023a), and NFTs (Zhou et al., 2023b), which impose permutation symmetry, and Monomial-NFNs (Tran et al., 2024) and MAGEP-NFN (Vo et al., 2025), which further impose scale symmetry. Such metanetworks are defined only for a fixed input width and depth, which prevents their application to networks of varying size. The second strategy views the input network as a graph, with neurons as nodes and weights as edges, and applies a message-passing GNN or transformer to it, as in GMNs (Lim et al., 2024) and neural graphs (Kofinas et al., 2024). ScaleGMN (Kalogeropoulos et al., 2024) extends GMNs to also impose scale symmetry. These models naturally apply to input networks of varying size, just as GNNs and transformers apply to graphs of varying size. A third strategy avoids operating on the full parameter space and instead represents networks through probes, gradients, or other derived quantities (Herrmann et al., 2024; Kahana et al., 2025; Heo et al., 2026; Gelberg et al., 2026), which reduces the need to model weight-space symmetries directly but requires suitable probe data and may miss information not exposed by the chosen inputs or derived signals. For a comprehensive review of weight-space symmetries and metanetworks, see Han et al. (2026).

In this work, we focus on the second approach. Here we concretely describe a simple GMN on MLP input networks, so that we can specify the proposed modifications in Section 3. [1]

[1] The extensions to other GMN variants from the literature are described in Appendix C. They involve some technicalities, but the underlying idea is the same throughout.

We first fix the notation for MLP input networks, that we will use throughout the paper. An L-layer MLP parameterizes a function from $\mathbb{R}^{d_{\text{in}}}$ to $\mathbb{R}^{d_{\text{out}}}$, mapping $x^{(0)} = x$ to $h^{(L+1)}$ by recursing

$$h^{(\ell)} = W^{(\ell)} x^{(\ell-1)} + b^{(\ell)} \in \mathbb{R}^{n_\ell}, \qquad x^{(\ell)} = \sigma\big(h^{(\ell)}\big) \in \mathbb{R}^{n_\ell}, \qquad \ell = 1, \dots, L+1, \tag{1}$$

where $n_0 = d_{\text{in}}$ and $n_{L+1} = d_{\text{out}}$, $(W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}})_{\ell=1}^{L+1}$ and $(b^{(\ell)} \in \mathbb{R}^{n_\ell})_{\ell=1}^{L+1}$ are the layer-wise weights and biases, and σ is the activation function applied elementwise. We write $\theta = \big((W^{(\ell)}, b^{(\ell)})\big)_{\ell=1}^{L+1}$ for the weights.

A GMN acting on an MLP input network is a message-passing GNN on the MLP's computation graph: given the weights θ, the graph has one node $v_i^{(\ell)}$ per neuron ($\ell = 0, \dots, L+1$, $i \in [n_\ell]$) with initial feature $h_i^{(\ell),[0]} = \psi_v\big(b_i^{(\ell)}\big)$ for $\ell \geq 1$ and a constant for $\ell = 0$, and one directed edge $v_j^{(\ell-1)} \to v_i^{(\ell)}$ with initial feature $e_{ji}^{(\ell),[0]} = \psi_e\big(W_{ij}^{(\ell)}\big)$. The model then runs T rounds of message passing: for $t = 0, \dots, T-1$, and $\ell = 0, \dots, L+1$,

$$\begin{aligned}
m_{j\to i}^{(\ell),[t]} &= \psi_m\big(e_{ji}^{(\ell),[t]}, h_j^{(\ell-1),[t]}\big), \qquad m_{i\leftarrow j}^{(\ell),[t]} = \psi_m^{\leftarrow}\big(e_{ij}^{(\ell+1),[t]}, h_j^{(\ell+1),[t]}\big), \\
h_i^{(\ell),[t+1]} &= \psi_h\big(h_i^{(\ell),[t]}, \sum_{j\in[n_{\ell-1}]} m_{j\to i}^{(\ell),[t]}, \sum_{j\in[n_{\ell+1}]} m_{i\leftarrow j}^{(\ell),[t]}\big), \\
e_{ji}^{(\ell),[t+1]} &= \psi_u\big(e_{ji}^{(\ell),[t]}, h_j^{(\ell-1),[t]}, h_i^{(\ell),[t]}\big).
\end{aligned} \tag{2}$$

The first aggregate is set to 0 at $\ell = 0$, and the second is set to 0 at $\ell = L+1$. Finally, the GMN reads out a graph-level prediction by pooling the final node features,

$$f^{\Theta}(\theta) = \rho\Big(\sum_{\ell=0}^{L+1} \sum_{i\in[n_\ell]} \psi_r\big(h_i^{(\ell),[T]}\big)\Big). \tag{3}$$

Each node $v_i^{(\ell)}$ aggregates a forward message $m_{j\to i}^{(\ell),[t]}$ along incoming edges and a backward message $m_{i\leftarrow j}^{(\ell),[t]}$ along outgoing ones. We call this the *bidirectional* GMN, and the restriction $\psi_m^{\leftarrow} = 0$, in which messages pass only along edge direction, gives the *forward* GMN. Components $\psi_v, \psi_e, \psi_m, \psi_m^{\leftarrow}, \psi_h, \psi_u, \psi_r, \rho$ are all shared MLPs, with parameters collected in Θ. Since Θ is independent of the width $(n_\ell)_{\ell=0}^{L+1}$, the GMN applies to MLPs of any width.

2.2 Any-dimensional learning, transferability, and size generalization

Our proposal of transferable GMNs is guided by the theory of any-dimensional learning and transferability in Levin et al. (2026). This theory studies models that accept inputs of any size, such as DeepSets, GNNs, and transformers. It asks when such a model keeps working well at sizes larger than those it was trained on. The theory identifies two properties a model can have, each defined relative to some structure on the input space that the target function itself respects. The first, *compatibility*, says the model must give the same output on two inputs that are equivalent, in the sense that the target function itself gives the same output on both. The second, *continuity*, says the model should vary continuously with respect to a prescribed metric that compares inputs across sizes, a metric under which the target itself varies continuously. Continuity is the stronger property: compatibility only transfers a prediction to larger inputs exactly equivalent to a training input. Continuity transfers it to every larger input merely *close* to one—the situation that arises in practice.

These are formalized through *consistent sequences*, which we briefly review since our later analysis relies on this formulation. Consider a family of vector spaces V_n, where V_n collects the objects of size n, together with linear embeddings $\varphi_{N,n} : V_n \hookrightarrow V_N$ for $n \leq N$ that send a size-n object to its size-N counterpart and thereby declare which objects of different sizes are equivalent.[2] Identifying each object with its larger counterparts gives the direct limit $V_\infty := \varinjlim_n V_n = \bigcup_n V_n$, a single space holding all sizes at once. Moreover, if each V_n carries a norm that respects the equivalence across sizes, that is, $\|\varphi_{N,n}(x)\|_{V_N} = \|x\|_{V_n}$ for $x \in V_n$, these norms glue into a norm $\|\cdot\|_{V_\infty}$ on V_∞ measuring distances across sizes. A sequence of functions $f_n \colon V_n \to \mathbb{R}$ is *compatible* if

[2]Each V_n also carries an action of a group G_n where $\mathsf{G}_n \hookrightarrow G_N$, and the embedding is group equivariant. For weight space applications, this models the permutation symmetries that prior works consider.

$f_N|_{V_n} = f_n$ for $n \le N$, equivalently if the f_n glue into a single $f_\infty : V_\infty \to \mathbb{R}$. The sequence f_n is *continuous* (or transferable) if it is compatible and f_∞ is *continuous* in $\|\cdot\|_{V_\infty}$. This formalizes the compatibility and continuity of models and targets mentioned above.

Weight spaces of varying width fit this template. Here V_n is the weight space of MLPs of width n: two weights are equivalent (respectively, close) when they represent the same (respectively, similar) functions. A GMN is an any-dimensional neural network accepting weights of different widths. In Section 3, we make the equivalence and the norm on weight space precise, and design a GMN that is compatible and continuous with respect to them. We give the full consistent-sequence formulation of the weight space in Appendix A.3.

2.3 μP, INFINITE-WIDTH LIMITS, AND TENSOR PROGRAMS

The relationship between the distribution of trained weights of different widths depends on the choice of parameterization (initialization and hyperparameter scalings with width). Table 7 summarizes two popular choices: the standard PyTorch default (SP) and μP. We emphasize that this choice affects only the *relationship* between widths, not the behavior at any single fixed width: each width-scaling in Table 7 carries a free multiplicative constant, which can always be chosen to make SP and μP agree at one chosen width.

The distributions of trained weights in the infinite-width limit, that is, as all hidden widths tend to infinity, are well studied. In our theoretical analysis in Section 5, we use the tensor-programs machinery of Littwin & Yang (2023), which treats the i.i.d. random initialization as the sole source of randomness, entering through the law of large numbers and the central limit theorem, to rigorously characterize the limiting distribution of activations and outputs at each training step. This analysis is what led to the derivation of μP (Yang & Hu, 2021), the unique parameterization achieving "optimal" feature-learning behavior. Under μP, networks of every width approximate this optimal infinite-width limit with error $O(1/\sqrt{n})$, which is why training dynamics look so similar across widths in practice. Separately, μP has become a popular choice because it enables zero-shot hyperparameter transfer across widths (Yang et al., 2021): hyperparameters tuned on a small network can be used directly on a larger one, since their optimal values stay close across widths (see Figure 8).

3 TRANSFERABLE GRAPH METANETWORKS

3.1 DUPLICATION-COMPATIBLE GRAPH METANETWORKS

As mentioned in Section 2.1, a metanetwork should be invariant to transformations of its input networks that leave the represented function unchanged. Prior work has taken permutation and scale symmetries, both acting within a single width, as its guiding design principle. To handle input networks of any width, we instead exploit a structure relating *different* widths that prior work has not used: weights of different widths can represent the same function.

Widening by duplicating weight entries. The simplest such transformation duplicates and rescales the weight entries (See Figure 1(b)). Recall the MLP weights $\theta = \big((W^{(\ell)}, b^{(\ell)})\big)_{\ell=1}^{L+1}$ from (1). Fix width multipliers $k_1, \dots, k_L \in \mathbb{N}$ and set $k_0 = k_{L+1} = 1$, so that only hidden layers grow, and define $\theta^{\uparrow} = \big((W^{\uparrow(\ell)}, b^{\uparrow(\ell)})\big)_{\ell=1}^{L+1}$ by

$$W^{\uparrow(\ell)} := W^{(\ell)} \otimes k_{\ell-1}^{-1} \mathbf{1}_{k_\ell} \mathbf{1}_{k_{\ell-1}}^{\top}, \qquad b^{\uparrow(\ell)} := b^{(\ell)} \otimes \mathbf{1}_{k_\ell}, \tag{4}$$

where $\otimes$ is the Kronecker product and $\mathbf{1}_k \in \mathbb{R}^k$ is the vector of ones.[3] Then $\theta^{\uparrow}$, of widths $(n_\ell k_\ell)_{\ell=0}^{L+1}$, represents the same function as θ. We call this the *duplication widening*. Indeed, by induction on ℓ the widened activations are $x^{(\ell)\uparrow} = x^{(\ell)} \otimes \mathbf{1}_{k_\ell}$ (the hidden neurons are duplicated), and at $\ell = L+1$, where $k_{L+1} = 1$, the two outputs agree. This weight-space structure has been studied in prior work on optimization landscapes (Fukumizu & Amari, 2000; Fukumizu et al., 2019) and on initializing wide networks from narrow ones (Chen et al., 2016; Ma et al., 2026), but to our knowledge it has not been exploited in the design of metanetworks. Also, this is not specific to simple MLPs but generalizes to almost all standard architectures (see Theorem 2.5 of Ma et al. (2026)).

[3] In PYTORCH: `W_up = W.repeat_interleave(k0, dim=0).repeat_interleave(k1, dim=1) / k1`, and `b_up = b.repeat_interleave(k0)`, where `k0` and `k1` are k_ℓ and $k_{\ell-1}$.

Duplication-compatible GMNs. Following the theoretical framework reviewed in Section 2.2, we call a metanetwork *duplication compatible* if it is invariant to this duplication widening, that is, if $f^\Theta(\theta^\uparrow) = f^\Theta(\theta)$ for every duplication widening (4) and every GMN parameter Θ. We show that three simple modifications of the GMN of Section 2.1 suffice to enforce it.

1. *Fan-in rescaled edge features*: initialize edge features using the rescaled weights $\overline{W}^{(\ell)} := n_{\ell-1} W^{(\ell)}$ rather than from $W^{(\ell)}$. This rescaling cancels the $k_{\ell-1}^{-1}$ factor in the duplication widening (4), so that the features exactly get duplicated when the GMN applies to widened weights.
2. *Mean aggregation*: replace the two sum-aggregations in (2) by mean-aggregations, $\frac{1}{n_{\ell-1}} \sum_{j \in [n_{\ell-1}]} m_{j \to i}^{(\ell),[t]}$ and $\frac{1}{n_{\ell+1}} \sum_{j \in [n_{\ell+1}]} m_{i \leftarrow j}^{(\ell),[t]}$. This is because summing over a wider neighborhood would scale up the incoming messages by the multiplicity of the duplication, whereas averaging returns the same value as before.
3. *Layer-wise mean readout*: replace the readout (3) by

$$f^\Theta(\theta) = \rho\big(r_1, \dots, r_L, (h_i^{(0),[T]})_{i \in [n_0]}, (h_i^{(L+1),[T]})_{i \in [n_{L+1}]}\big), \quad r_\ell := \frac{1}{n_\ell} \sum_{i \in [n_\ell]} \psi_r\big(h_i^{(\ell),[T]}\big).$$

   In the hidden layers, taking layer-wise mean over node features is invariant to duplication. Input and output node features may be listed coordinatewise because n_0 and n_{L+1} do not grow.

Proposition 1 (Duplication compatibility)**.** *The GMN with the three modifications above is duplication compatible.*

None of the three modifications reduces expressivity at a fixed width. Rescaling the edge features and averaging instead of summing only rescale intermediate quantities by constants depending on the width, which the learned maps can absorb. The layer-wise mean readout retains strictly more information than the single global sum it replaces.

3.2 Matrix-product Graph Metanetworks

The duplication-compatible GMN has two shortcomings that hinder size generalization.

Problem 1: continuity in the wrong norm. Compatibility only ensures generalization to wide weights *exactly* equivalent to a training weight. As discussed in Section 2.2, generalizing to weights that are merely *nearby* additionally requires the metanetwork to be continuous, that is, robust to small perturbations: weights representing nearly the same function should receive nearly the same output. Which perturbations count as small is not a free choice, it is dictated by the target.

To make this precise, let $V_n := \bigoplus_{\ell=1}^{L+1} (\mathbb{R}^{n_\ell \times n_{\ell-1}} \oplus \mathbb{R}^{n_\ell})$ be the weight space of width $n = (n_1, \dots, n_L)$, embedded in V_N for $n_\ell \mid N_\ell$ by the duplication widening $\theta \mapsto \theta^\uparrow$ (4). This makes (V_n) a consistent sequence, in which equivalent weights of different widths represent the same function. On each V_n, we consider two norms: the *normalized spectral norm* $\|\cdot\|_{V_n,\mathrm{op}}$ and the *normalized* ℓ_1 *norm* $\|\cdot\|_{V_n,1}$

$$\|\theta\|_{V_n,\mathrm{op}} = \sum_{\ell=1}^{L+1} \Big(\sqrt{\tfrac{n_{\ell-1}}{n_\ell}} \Big\| W^{(\ell)} \Big\| + \tfrac{1}{\sqrt{n_\ell}} \Big\| b^{(\ell)} \Big\|_2 \Big), \quad \|\theta\|_{V_n,1} = \sum_{\ell=1}^{L+1} \Big(\tfrac{\big\|\overline{W}^{(\ell)}\big\|_1}{n_\ell n_{\ell-1}} + \tfrac{\|b^{(\ell)}\|_1}{n_\ell} \Big), \quad (5)$$

where $\|\cdot\|$ is the spectral norm, $\|\cdot\|_1$ the entrywise ℓ_1 norm, and $\|\cdot\|_2$ the Euclidean norm. Both norms satisfy $\|\theta\|_{V_n} = \big\|\theta^\uparrow\big\|_{V_N}$ for every duplication widening (4), so they are compatible with the equivalence across widths. Hence, they induce norms $\|\cdot\|_{V_\infty,\mathrm{op}}$ and $\|\cdot\|_{V_\infty,1}$ on the limit space V_∞, defining two different notions of distance between weights of different widths. Appendix A.3 carries this construction out in full and explains what each norm measures.

The target and the duplication-compatible GMN turn out to respect different ones of the two.

Proposition 2 (Misalignment)**.** *For the input MLP, let its input space $X \subseteq \mathbb{R}^{d_\mathrm{in}}$ be nonempty and compact, and let its activation function σ be Lipschitz. Equip the space of continuous functions $\mathcal{C}(X, \mathbb{R}^{d_\mathrm{out}})$ with the uniform norm.*

1. *Let $\mathcal{R}\colon V_\infty \to \mathcal{C}(X, \mathbb{R}^{d_{\text{out}}})$ be the realization map that sends the weights to the function they represent. Then $\mathcal{R}$ is Lipschitz on every bounded ball of $\|\cdot\|_{V_\infty,\text{op}}$ with a width-independent constant. However, if σ is unbounded, then $\mathcal{R}$ can be discontinuous in $\|\cdot\|_{V_\infty,1}$.*

2. *The duplication-compatible GMN with Lipschitz MLP components is Lipschitz in $\|\cdot\|_{V_\infty,1}$. However, if the input MLP has more than one layer, there exist duplication-compatible GMNs that are discontinuous in $\|\cdot\|_{V_\infty,\text{op}}$.*

The implication of this misalignment is the following: Weights close in $\|\cdot\|_{V_\infty,\text{op}}$ represent similar functions, hence have similar targets, yet the duplication-compatible GMN may map them to very different outputs; weights close in $\|\cdot\|_{V_\infty,1}$ do receive similar outputs from it, but may represent very different functions. Either way, the architecture itself puts no constraint on its predictions at unseen widths. Such misalignment was observed empirically to hinder size generalization across a range of tasks for any-dimensional models (Levin et al., 2026), so we expect the same to happen here.

Problem 2: invariance to the simplest widening only. The duplication widening (4) is not the only function-preserving widening. Partition $W^{\uparrow(\ell)}$ into blocks $W^{\uparrow(\ell)}_{ij} \in \mathbb{R}^{k_\ell \times k_{\ell-1}}$ for $i \in [n_\ell], j \in [n_{\ell-1}]$, and ask only that these blocks satisfy, for each ℓ,

$$W^{\uparrow(\ell)}_{ij}\mathbf{1}_{k_{\ell-1}} = W^{(\ell)}_{ij}\mathbf{1}_{k_\ell} \quad b^{\uparrow(\ell)} = b^{(\ell)} \otimes \mathbf{1}_{k_\ell}. \tag{6}$$

In words, every row of the block $W^{\uparrow(\ell)}_{ij}$ sums to $W^{(\ell)}_{ij}$: the copied weights can be redistributed across rows of the block, as illustrated in Figure 1(c). Then $\theta^\uparrow$ again represents the same function as θ. We call these *block-wise widenings*, and the previous duplication widening (4) is the special case in which every block is constant. Indeed, each copy of neuron i still collects the whole of the original weight from the copies of neuron j, but is free to collect it from them in any proportion. Appendix A.1 derives (6) as exactly the class of widenings whose hidden neurons are copies of the narrow network's (Figure 1(a)), so it is the most general of its kind, and Appendix B.2 proves that the duplication-compatible GMN is not invariant to it.

Matrix-product GMNs. Both shortcomings share one cause: the hidden weights are processed entrywise by $\psi_m, \psi_m^{\leftarrow}$ in (2). Removing this freedom remedies both at once. We modify the duplication-compatible GMN to define the *bidirectional matrix-product GMN*: drop all edge features beyond $\overline{W}^{(\ell)}_{ij}$, and let each weight act on the node feature by plain multiplication:

$$m^{(\ell),[t]}_{j\to i} = \overline{W}^{(\ell)}_{ij}\, A^{\text{in}}_\ell h^{(\ell-1),[t]}_j, \qquad m^{(\ell),[t]}_{i\leftarrow j} = \overline{W}^{(\ell+1)}_{ji}\, A^{\text{out}}_\ell h^{(\ell+1),[t]}_j, \tag{7}$$

for learnable matrices $A^{\text{in}}_\ell, A^{\text{out}}_\ell \in \mathbb{R}^{d_h \times d_h}$. The *forward matrix-product GMN* is the case $A^{\text{out}}_\ell = 0$. Equivalently, stacking the layer-ℓ node features as the rows of $H^{(\ell),[t]} \in \mathbb{R}^{n_\ell \times d_h}$, the mean aggregates of (7) are $W^{(\ell)}H^{(\ell-1),[t]}A^{\text{in}}_\ell$ and $\frac{n_\ell}{n_{\ell+1}}(W^{(\ell+1)})^\top H^{(\ell+1),[t]}A^{\text{out}}_\ell$: the weights $(W^{(\ell)})$ now enter only via matrix multiplication. This one change resolves both shortcomings: matrix multiplication is exactly the operation controlled by the spectral norm, resolving Problem 1's misalignment, and exactly the operation left invariant by block-wise widenings (6), resolving Problem 2.

Proposition 3. *1. The matrix-product GMN with Lipschitz MLP components is Lipschitz on every bounded ball of $\|\cdot\|_{V_\infty,\text{op}}$ with a width-independent constant.*

2. *The forward matrix-product GMN satisfies $f^\Theta(\theta^\uparrow) = f^\Theta(\theta)$ for every block-wise widening* (6)*. The bidirectional one satisfies it for those block-wise widenings whose blocks additionally have equal column sums, $\mathbf{1}_{k_\ell}^\top W^{\uparrow(\ell)}_{ij} = \frac{k_\ell}{k_{\ell-1}} W^{(\ell)}_{ij}\mathbf{1}_{k_{\ell-1}}^\top$ for all i, j, ℓ.*

The matrix-product GMN is a strict restriction of the duplication-compatible GMN. It is presumably less expressive at a fixed width, but we do not observe any performance loss in all experiments.

4 Experiments

We empirically study the size generalization of our transferable GMNs, comparing three models: a baseline bidirectional GMN with positional encoding and residual connections, using the implementation from (Kalogeropoulos et al., 2024); our duplication-compatible GMN; and our matrix-product

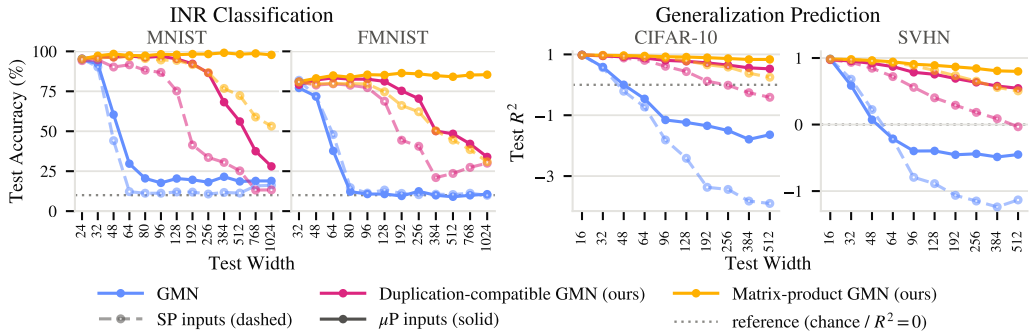


Figure 2: Size generalization with independently trained weights, comparing the baseline GMN, the duplication-compatible GMN, and the matrix-product GMN on SP- and μP-trained input networks. Each panel trains at the smallest width and evaluates at the sequence of larger widths shown. Columns 1–2: test accuracy on the INR classification tasks, with the random-guessing baseline. Columns 3–4: test R^2 against ground-truth accuracies on the CNN generalization prediction task, with $R^2 = 0$ (mean-accuracy prediction) as baseline.

GMN, the latter two obtained from that baseline via the proposed modifications. We also apply the modifications to other baselines, including the forward and bidirectional ScaleGMN, with those results and all other experiment details in Appendix D.

We experiment on two tasks, each with two datasets of different difficulty, all adapted from (Kalogeropoulos et al., 2024) and used repeatedly in prior work (Zhou et al., 2023a; 2024; Unterthiner et al., 2020; Lim et al., 2024; Kofinas et al., 2024): (1) *INR classification*: given the weights of an MLP fitted to an MNIST (LeCun et al., 1998) or FMNIST (Xiao et al., 2017) image, predict the image class and report accuracy; and (2) *generalization prediction*: given the weights of a CNN trained on CIFAR-10 (Krizhevsky et al., 2009) or SVHN (Netzer et al., 2011), predict its test accuracy and report the R^2 score. We further report other metrics, such as Kendall's τ correlation (Kendall et al., 1938), in Appendix D. In all experiments the GMN is trained at one small width, with hyperparameters selected on the validation metric at that width only, and evaluated at a range of larger widths, constructed in two ways that we discuss separately below.

Size generalization on hand-crafted widenings. As a sanity check, we widen the training-set networks by hand via the duplication widening (4) and the block-wise widening (6), both of which preserve the represented function exactly. This is a controlled setting, since our modified GMNs are designed to capture these equivalences and the baseline GMN is not. The results match the theory exactly and are given in Appendix D.2.

Size generalization on independently trained weights. The more realistic scenario is when the weights at each width are trained independently from random initialization, where it is unclear a priori whether they are even approximately a block-wise widening (6) of the training-width networks, and hence whether our modifications help at all. Figure 2 shows the results, and Figure 5 those for the other baseline GMN models. The observations below are consistent across all tasks, datasets, and GMN models:

1. On every dataset, performance improves in the order GMN, duplication-compatible GMN, matrix-product GMN, confirming that both modifications help, and echoing the examples of Levin et al. (2026) in which both compatibility and continuity matter.
2. For every model, size generalization is much better on μP-trained inputs than on SP-trained ones, even though μP is calibrated to coincide with SP at the training width, so the two share the same training distribution. Intuitively, μP-trained networks have similar training dynamics across widths, so small widths inform predictions at large ones; under SP the trained weights vary more, making size generalization inherently harder.
3. Combining the matrix-product GMN with μP inputs gives an almost flat test score across all larger widths, which we justify theoretically next.

5 THEORY OF SIZE GENERALIZATION ON TRAINED WEIGHTS

In Figure 2, the test score stays nearly constant across widths only for the matrix-product GMN on μP-trained inputs, degrading visibly as soon as either ingredient is removed. We now prove size-generalization guarantees that explain this.

We first specify the data distribution: one law μ_n per width $n = (n_\ell)_{\ell=1}^L$, all produced by a single training rule, with μ_n the law of the trained weights $\theta_n = \mathsf{Train}_n(\Gamma, S, \xi_n) \in V_n$. Here Train_n is the training procedure encoding the optimization algorithm and the parameterization, Γ collects the hyperparameters, ξ_n is the random initialization, and S is the remaining randomness coming from mini-batching, etc. We assume Γ, S, ξ_n are mutually independent random variables, and take Γ and S to be independent of width, so μ_n depends on the width only through Train_n and ξ_n.

Now fix a GMN f^Θ, a target function f^*, and a loss ℓ, and write $\mathcal{R}_m(f^\Theta) = \mathbb{E}_{\theta\sim\mu_m}\, \ell(f^\Theta(\theta), f^*(\theta))$ for the expected loss at width m. If f^Θ is trained at width n and tested on all widths $m \geq n$, its *size-generalization gap* $\sup_{m\geq n} \left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right|$ is the worst discrepancy between its in-distribution and out-of-distribution test error, exactly what the flatness of a curve in Figure 2 shows. We say that f^Θ *size generalizes* if the gap vanishes as $n \to \infty$. We will give two guarantees below. Both control this gap through the Tensor Programs framework of Littwin & Yang (2023), which rigorously characterizes the infinite-width limit of the training dynamics, and hence of μ_n. We state them informally here, with full details given in Appendix E.

The first guarantee covers 1-layer input MLPs only, but comes with a clean geometric interpretation. We have shown in Proposition 3(i) that the matrix-product GMN parameterizes a function on V_∞ that is continuous with respect to d_∞^{op}, the distance induced by the normalized spectral norm $\|\cdot\|_{V_\infty,\mathrm{op}}$ (5) after quotienting by hidden-unit permutations. So if the trained weights θ_n of different widths are close in d_∞^{op} with high probability, performance transfers across widths: intuitively, the model generalizes to larger inputs that are "similar" to the training-width inputs, with d_∞^{op} capturing the notion of similarity that the GMN respects. This is exactly what μP delivers: throughout training, the weights attached to the n hidden neurons behave like n i.i.d. draws from a single width-independent law, forcing the trained weights of different widths to be close in d_∞^{op}.

Theorem 4 (Informal)**.** *Let μ_n be the law of the trained weights of a 1-layer MLP under μP. Then $\sup_{m\geq n} W_1(\mu_n, \mu_m; d_\infty^{\mathrm{op}}) \to 0$ as $n \to \infty$, where W_1 is the Wasserstein-1 distance with respect to d_∞^{op}. Consequently, if the target is a Lipschitz functional of the represented function and the loss is Lipschitz, then $\sup_{m\geq n} \left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right| \to 0$ for every duplication-compatible GMN f^Θ with Lipschitz components, matrix-product ones included.*

This covers all duplication-compatible GMNs because for 1-layer MLP $\|\cdot\|_{V_\infty,1}$ is dominated by the $\|\cdot\|_{V_\infty,\mathrm{op}}$ (Proposition 10), so Proposition 2(ii) already gives Lipschitzness in d_∞^{op}: for 1-layer inputs there is no misalignment, and the weaker model suffices. The theorem also explains why SP fails: it initializes the weights so that $\|\theta_n\|_{V_n,\mathrm{op}} = \Theta(\sqrt{n})$, which already at initialization forces $W_1(\mu_n, \mu_{2n}; d_\infty^{\mathrm{op}}) \to \infty$. The weights drift apart as width increases, rather than settling to a common limit, leaving the GMN's predictions at unseen widths unconstrained.

Beyond one hidden layer, we give a second guarantee that works at any depth. Here we instead compare the laws of the trained weights of different widths, μ_n and μ_m, through the integral probability metric generated by the GMN losses themselves, $d_{\mathcal{A}}(\mu_n, \mu_m) := \sup_{\Theta\in\mathcal{A}} \left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right|$ over a compact set $\mathcal{A}$ of GMN parameters, which dominates the gap by construction. This lets us bypass the limitation of the d_∞^{op}-distance, which does not fully capture the notion of similarity that the GMN respects at depth $L > 1$. However, this guarantee makes a stronger assumption on the target: we state it here for targets exactly representable by the GMN, but it also holds under less restrictive assumptions, as discussed in Appendix E.3.

Theorem 5 (Informal)**.** *Let μ_n be the law of the trained weights of a fixed-depth MLP under μP, and let f^Θ be the matrix-product GMN. Suppose the loss is Lipschitz and the target is representable by such a GMN. Then $\sup_{m\geq n} d_{\mathcal{A}}(\mu_n, \mu_m) \to 0$ as $n \to \infty$, and hence the size-generalization gap vanishes uniformly over $\Theta \in \mathcal{A}$.*

Taken together, the two guarantees capture the interplay of the GMN architecture and the data distribution in size generalization. Under SP our modifications still help empirically, so we expect the mechanism to reach beyond the regime our theory covers.

AI USE STATEMENT

Generative AI tools were used extensively in this work, under the authors' direction and review throughout. Their use was iterative rather than one-shot: nothing was adopted as returned from a single prompt, and every AI-assisted artifact went through repeated rounds of author-specified goals, assistant drafts, and author correction.

We used generative AI to *implement methods*, to *generate synthetic datasets*, to *clean and reformat datasets*, to *design or provide feedback on research methodology or experiments*, to *provide critical ingredients for proving mathematical claims*, to *assist in the writing of proofs*, to *propose or refine hypotheses*, and, in the limited sense described below, to *formulate mathematical claims*. We did not use generative AI to *help develop theoretical models or conceptual frameworks*, or to *interpret results*. The remaining required-disclosure tasks (*assisting with translation* and *support for qualitative and thematic data analysis*) are not applicable to this work. We also used generative AI to *edit the paper to improve readability* and to *summarize or analyze existing literature*. We have reviewed all AI-assisted work.

The main text was written by the authors; the generative AI polished, proofread, and typeset it. In the theoretical part the main ideas are the authors' own, including but not limited to, the application of the transferability framework to the GMN setting, the weight-space equivalences and metrics, the design of transferable GMN modifications, and the use of Tensor Programs to analyze the distribution of trained weights. The generative AI wrote up arguments the authors had already settled on, searched them for errors and missing hypotheses, and checked notation and cross-references, over several rounds of author correction per argument. Most theorem statements and proof strategies are likewise the authors' own, with the AI assistant extending them to more general settings and carrying out mundane calculations. We do highlight that the idea behind Theorem 5 was proposed by the generative AI, and reached its final form through repeated exchanges in which the authors fixed the setting and corrected the argument. All AI-generated statements and proofs were substantially revised and reorganized by the authors for readability, clarity and correctness. They were verified independently of the AI assistant's own checks and, wherever possible, validated numerically. The experimental codebase was written with the AI coding assistant under the same process of author specification, inspection, and debugging; all results are logged and audited by the authors, with the AI tools helping to tabulate and visualize them.

We take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All data used in this work are publicly available benchmark datasets or model checkpoints that we trained ourselves on these datasets. None of it involves human subjects, personally identifiable information, or sensitive attributes. This is a fundamental-research contribution to metanetwork methodology, and we do not foresee a direct pathway from our results to a harmful application.

REPRODUCIBILITY STATEMENT

Full statements of our theoretical results, their assumptions, and their proofs are given in Appendix B, with further details in Appendix E. Experimental details are given in Appendix D. We will release the code in the camera-ready version.

REFERENCES

Guy Bar-Shalom, Ami Tavory, Itay Evron, Maya Bechler-Speicher, Ido Guy, and Haggai Maron. A graph meta-network for learning on kolmogorov-arnold networks. *arXiv preprint arXiv:2602.16316*, 2026.

Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *International Conference on Machine Learning*, pp. 837–851. PMLR, 2021.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, 33:13260–13271, 2020.

Adir Dayan, Yam Eitan, and Haggai Maron. On the expressive power of permutation-equivariant weight-space networks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gQXkbQSZWS.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5694–5725. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/dupont22a.html.

Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*. John Wiley & Sons, 2009.

Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327, 2000.

Kenji Fukumizu, Shoichiro Yamaguchi, Yoh-ichi Mototake, and Mirai Tanaka. Semi-flat minima and saddle points by embedding neural networks to overparameterization. *Advances in neural information processing systems*, 32, 2019.

Yoav Gelberg, Yam Eitan, Aviv Navon, Aviv Shamsian, Theo Putterman, Michael Bronstein, and Haggai Maron. Gradmetanet: An equivariant architecture for learning on gradients. *Advances in Neural Information Processing Systems*, 38:154115–154158, 2026.

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.

Eunwoo Heo, Kyeongkook Seo, and Jaejun Yoo. What linear probes miss: Multi-view probing for weight-space learning. *arXiv preprint arXiv:2605.23410*, 2026.

Vincent Herrmann, Francesco Faccio, and Jürgen Schmidhuber. Learning useful representations of recurrent neural network weight matrices. *arXiv preprint arXiv:2403.11998*, 2024.

Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep linear probe generators for weight space learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XoYdD3m0mv.

Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. *Advances in neural information processing systems*, 37:106800–106840, 2024.

Maurice G Kendall et al. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. Convergence and stability of graph convolutional networks on large random graphs. *Advances in Neural Information Processing Systems*, 33:21512–21523, 2020.

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph Neural Networks for Learning Equivariant Representations of Neural Networks. In *12th International Conference on Learning Representations (ICLR)*, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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.

Ron Levie. A graphon-signal analysis of graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Eitan Levin, Yuxin Ma, Mateo Dìaz, and Soledad Villar. On transferring transferability: Towards a theory for size generalization, 2026. URL https://arxiv.org/abs/2505.23599.

Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ijK5hyxs0n.

Etai Littwin and Greg Yang. Tensor programs IVb: Adaptive optimization in the infinite-width limit. In *The Eleventh International Conference on Learning Representations*, 2023.

Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OoOIW-3uadi.

Yuxin Ma, Nan Chen, Mateo Diaz Diaz, Soufiane Hayou, Dmitriy Kunisky, and Soledad Villar. μpscaling small models: Principled warm starts and hyperparameter transfer. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bAqpZRzlTg.

Sohir Maskey, Ron Levie, and Gitta Kutyniok. Transferability of graph neural networks: an extended graphon approach. *Applied and Computational Harmonic Analysis*, 63:48–83, 2023.

Nikhil Mehta, Jonathan Lorraine, Steve Masson, Ramanathan Arunachalam, Zaid Pervaiz Bhat, James Lucas, and Arun George Zachariah. Improving hyperparameter optimization with checkpointed model weights. In *European Conference on Computer Vision*, pp. 75–96. Springer, 2024.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in applied probability*, 29(2):429–443, 1997.

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, 2023a.

Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. *arXiv preprint arXiv:2310.13397*, 2023b.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17020–17031, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/12bcd658ef0a540cabc36cdf2b1046fd-Abstract.html.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. *Advances in Neural Information Processing Systems*, 34:16481–16493, 2021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Viet-Hoang Tran, Thieu Vo, Tho Tran-Huu, An T Nguyen, and Tan Nguyen. Monomial matrix group equivariant neural functional networks. *Advances in Neural Information Processing Systems*, 37: 48628–48665, 2024.

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

Mauricio Velasco, Kaiying O'Hare, Bernardo Rychtenberg, and Soledad Villar. Graph neural networks and non-commuting operators. *Advances in neural information processing systems*, 37: 95662–95691, 2024.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=eTDgECpQ2I.

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

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in neural information processing systems*, 32, 2019.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 18–24 Jul 2021.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11975–11986. PMLR, 2021. URL https://proceedings.mlr.press/v139/yehudai21a.html.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in neural information processing systems*, 36:24966–24992, 2023a.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. *Advances in neural information processing systems*, 36:77485–77502, 2023b.

Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. *Advances in neural information processing systems*, 37:104754–104775, 2024.

CONTENTS

A Structure of the weight space **15**

A.1 Widening by duplicating neurons 15

A.2 Widening by zero-padding neurons 17

A.3 The weight space as a consistent sequence 18

B Theoretical properties of transferable GMNs **22**

B.1 Duplication-compatibility 22

B.2 Compatibility with block-wise widening 23

B.3 Task-model misalignment 24

B.4 Spectral continuity of matrix-product GMNs 28

B.5 Zero-padding compatibility 29

B.5.1 Lipschitz sensitivity of GMNs to their parameters. 31

C Implementations of transferable GMNs **34**

C.1 Modifications on the GMN of Lim et al. (2024) 35

C.2 Modifications on the NG-GNN of Kofinas et al. (2024) 38

C.3 Modifications on the ScaleGMN of Kalogeropoulos et al. (2024) 40

D Experiments **43**

D.1 Verification of GMN compatibility and continuity 43

D.2 Size generalization on hand-crafted widenings 44

D.3 Size generalization on independently trained weights 47

D.3.1 Dataset generation 47

D.3.2 Evaluation and results 48

E Theory of size generalization on independently trained weights **50**

E.1 Preliminary: Tensor Programs and μP 50

E.2 Proof of Theorem 4 57

E.2.1 A high-level explanation 57

E.2.2 Proofs of the 1-layer results 59

E.2.3 Obstruction on MLP with larger depths 62

E.3 Proof of Theorem 5 62

A STRUCTURE OF THE WEIGHT SPACE

Section 3 builds the transferable GMN to respect the following two weight space structures: (1) invariance to transformations that form wider weights representing the same function, and (2) continuity with respect to a distance that compares weights of different widths. We now discuss these structures in more detail, showing that they are not arbitrary choices but rather the natural ones to consider.

We first discuss in Appendix A.1 the duplication widening (4) and the block-wise widening (6) from the main text. Then, in Appendix A.2, we explore another natural transformation that also produces a wider network representing the same function. We further explain why we do not consider this in our design of the transferable GMN. Finally, in Appendix A.3, we use the consistent-sequence framework of Levin et al. (2026) (reviewed in Section 2.2) to model the weight space structure across different widths, and give the details of the norms defined on the weight space discussed in Section 3.

A common derivation. Both Appendix A.1 and Appendix A.2 instantiate one recipe, which we set up once here. Fix an MLP (1) with weights θ and widths $(n_\ell)_{\ell=0}^{L+1}$, and read it as a chain of maps,

$$\mathbb{R}^{n_0} \xrightarrow{W^{(1)}} \mathbb{R}^{n_1} \xrightarrow{+b^{(1)}} \mathbb{R}^{n_1} \xrightarrow{\sigma} \mathbb{R}^{n_1} \xrightarrow{W^{(2)}} \cdots \xrightarrow{W^{(L+1)}} \mathbb{R}^{n_{L+1}} \xrightarrow{+b^{(L+1)}} \mathbb{R}^{n_{L+1}}.$$

Fix target widths $(N_\ell)_{\ell=0}^{L+1}$ with $N_0 = n_0$ and $N_{L+1} = n_{L+1}$, so that only the hidden layers grow, together with injective linear maps

$$\varphi_\ell \colon \mathbb{R}^{n_\ell} \to \mathbb{R}^{N_\ell}, \qquad \ell = 0, \dots, L+1, \qquad \varphi_0 = I_{n_0}, \quad \varphi_{L+1} = I_{n_{L+1}},$$

one per layer. The maps φ_ℓ give the prescription for how a width-n_ℓ activation vector is to be written at width N_ℓ, and one can think of them as defining a structure-preserving embedding of the entire narrow network into the wide one. We ask: what are the wide weights $\theta^\uparrow = (W^{\uparrow(\ell)}, b^{\uparrow(\ell)})$ of widths $(N_\ell)_{\ell=0}^{L+1}$ so that the diagram

$$\begin{array}{ccccccc}
\mathbb{R}^{n_{\ell-1}} & \xrightarrow{W^{(\ell)}} & \mathbb{R}^{n_\ell} & \xrightarrow{+b^{(\ell)}} & \mathbb{R}^{n_\ell} & \xrightarrow{\sigma} & \mathbb{R}^{n_\ell} \\
\downarrow{\scriptstyle \varphi_{\ell-1}} & & \downarrow{\scriptstyle \varphi_\ell} & & \downarrow{\scriptstyle \varphi_\ell} & & \downarrow{\scriptstyle \varphi_\ell} \\
\mathbb{R}^{N_{\ell-1}} & \xrightarrow[W^{\uparrow(\ell)}]{} & \mathbb{R}^{N_\ell} & \xrightarrow[+b^{\uparrow(\ell)}]{} & \mathbb{R}^{N_\ell} & \xrightarrow[\sigma]{} & \mathbb{R}^{N_\ell}
\end{array}$$

to commute for every ℓ. That is,

$$W^{\uparrow(\ell)} \varphi_{\ell-1} = \varphi_\ell W^{(\ell)}, \qquad b^{\uparrow(\ell)} = \varphi_\ell b^{(\ell)}, \qquad \sigma \circ \varphi_\ell = \varphi_\ell \circ \sigma. \tag{8}$$

This asks for the wide network to match the narrow one at each layer ℓ (via the map φ_ℓ), that is, $h^{\uparrow(\ell)} = \varphi_\ell h^{(\ell)}$ and $x^{(\ell)\uparrow} = \varphi_\ell x^{(\ell)}$ for every ℓ. In particular, this asks for more than $\mathcal{R}(\theta^\uparrow) = \mathcal{R}(\theta)$, which is given by the case $\ell = L+1$.

We consider two natural choices of φ_ℓ. In Appendix A.1 we take φ_ℓ to duplicate the coordinates; in Appendix A.2 we take it to zero-pad them.

A.1 WIDENING BY DUPLICATING NEURONS

We first consider the case where the wide network reproduces the narrow network's forward pass with each hidden neuron replaced by several copies of itself. In other words, let

$$\varphi_\ell \colon \mathbb{R}^{n_\ell} \to \mathbb{R}^{N_\ell}, \qquad x \mapsto x \otimes \mathbf{1}_{k_\ell}. \tag{9}$$

In words, φ_ℓ duplicates the ith coordinate of x exactly k_ℓ times. We ask what the corresponding conditions on the wide weights $\theta^\uparrow$ are so that (8) holds.

Two of the three conditions of (8) are immediate. The bias condition reads $b^{\uparrow(\ell)} = b^{(\ell)} \otimes \mathbf{1}_{k_\ell}$. The activation condition holds for *every* elementwise activation σ. The weight condition is the content of the following proposition.

Proposition 6. *Let φ_ℓ be as in* (9) *and partition $W^{\uparrow(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ into blocks $W^{\uparrow(\ell)}_{ij} \in \mathbb{R}^{k_\ell \times k_{\ell-1}}$, $i \in [n_\ell]$ and $j \in [n_{\ell-1}]$. Then, for each ℓ, $W^{\uparrow(\ell)}\varphi_{\ell-1} = \varphi_\ell W^{(\ell)}$ if and only if*

$$W^{\uparrow(\ell)}_{ij}\mathbf{1}_{k_{\ell-1}} = W^{(\ell)}_{ij}\mathbf{1}_{k_\ell} \qquad \text{for all } i, j.$$

Proof. For the $\Leftarrow$ direction, let $x \in \mathbb{R}^{n_{\ell-1}}$. The jth block of $\varphi_{\ell-1}x$ is $x_j\mathbf{1}_{k_{\ell-1}}$, so the ith block of $W^{\uparrow(\ell)}\varphi_{\ell-1}x$ is, by block matrix multiplication,

$$\sum_{j\in[n_{\ell-1}]} W^{\uparrow(\ell)}_{ij}\left(x_j\mathbf{1}_{k_{\ell-1}}\right) = \sum_{j\in[n_{\ell-1}]} x_j W^{(\ell)}_{ij}\mathbf{1}_{k_\ell} = \left(W^{(\ell)}x\right)_i \mathbf{1}_{k_\ell},$$

where the first equality is the hypothesis and the second is the definition of the matrix–vector product. This is the ith block of $\varphi_\ell W^{(\ell)}x$, and i was arbitrary.

For the $\Rightarrow$ direction, evaluate the identity at the standard basis vector $x = e_j \in \mathbb{R}^{n_{\ell-1}}$, for which $\varphi_{\ell-1}e_j$ has $\mathbf{1}_{k_{\ell-1}}$ in its jth block and zeros elsewhere. Reading off the ith output block of both sides gives $W^{\uparrow(\ell)}_{ij}\mathbf{1}_{k_{\ell-1}} = W^{(\ell)}_{ij}\mathbf{1}_{k_\ell}$, for every pair (i, j). □

Together with the bias condition, Proposition 6 proves that the block-wise widening (6) of the main text preserves the represented function. Moreover, since it is proved with "if and only if" conditions, it gives the most general widening operation that does so while keeping the hidden neurons of the wide network as copies of the narrow network's at each layer.

This can be interpreted in terms of the flow of information on the computation graph of the network. In the narrow network, neuron i of layer ℓ draws the amount $W^{(\ell)}_{ij}$ along its edge from neuron j of layer $\ell - 1$. In the wide network, each of the k_ℓ copies of neuron i must draw that same total $W^{(\ell)}_{ij}$ from the $k_{\ell-1}$ copies of neuron j *taken together*, but it is free to split the total among those copies however it likes, and different copies of i may split it differently. Only the total matters because the copies of neuron j all carry the same activation, so the incoming contribution depends on the weights only through their sum. At one extreme a copy of i may route the whole of $W^{(\ell)}_{ij}$ through a single copy of j and ignore the rest; at the other it may split it evenly. These extremes single out the parallel widening and the duplication widening that we discuss below.

Two more rigid subfamilies. The block-wise widening (6) is flexible but not very structured. It is worth naming the successively more rigid choices inside the block-wise widening (6), since this is how the duplication widening (4) of the main text arises.

Proposition 7. *Let $P^{(\ell)} \in \mathbb{R}^{k_\ell \times k_{\ell-1}}$ and set $W^{\uparrow(\ell)} := W^{(\ell)} \otimes P^{(\ell)}$. Then for each ℓ, $W^{\uparrow(\ell)}\varphi_{\ell-1} = \varphi_\ell W^{(\ell)}$ holds for every $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ if and only if $P^{(\ell)}\mathbf{1}_{k_{\ell-1}} = \mathbf{1}_{k_\ell}$, that is, if and only if every row of $P^{(\ell)}$ sums to one.*

Proof. By the mixed-product property of the Kronecker product, for every $x \in \mathbb{R}^{n_{\ell-1}}$,

$$\left(W^{(\ell)} \otimes P^{(\ell)}\right)\left(x \otimes \mathbf{1}_{k_{\ell-1}}\right) = \left(W^{(\ell)}x\right) \otimes \left(P^{(\ell)}\mathbf{1}_{k_{\ell-1}}\right),$$

whereas $\varphi_\ell W^{(\ell)}x = \left(W^{(\ell)}x\right)\otimes\mathbf{1}_{k_\ell}$. If $P^{(\ell)}\mathbf{1}_{k_{\ell-1}} = \mathbf{1}_{k_\ell}$ the two agree, which gives $\Leftarrow$. Conversely, choose $W^{(\ell)}$ and x with $u := W^{(\ell)}x \neq 0$; then $u \otimes \left(P^{(\ell)}\mathbf{1}_{k_{\ell-1}} - \mathbf{1}_{k_\ell}\right) = 0$ with $u \neq 0$ forces $P^{(\ell)}\mathbf{1}_{k_{\ell-1}} = \mathbf{1}_{k_\ell}$. □

We call this *Kronecker widening*. The condition $P^{(\ell)}\mathbf{1} = \mathbf{1}$ says that the rows of $P^{(\ell)}$ sum to one; if $P^{(\ell)}$ is in addition entrywise nonnegative it is row-stochastic, and the split of each total $W^{(\ell)}_{ij}$ among the copies of j is a genuine convex combination.

Two instances are worth having in mind.

1. When adjacent hidden-layer multipliers agree, taking $P^{(\ell)} = I_k$ sends each copy of a neuron only to the corresponding copy in the next layer; with the analogous boundary choices at the first and last layers, this gives k parallel, non-interacting copies of the narrow network whose outputs are recombined by the last layer. We call this the *parallel widening*.
2. Taking $P^{(\ell)} = k_{\ell-1}^{-1}\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top$ gives exactly the duplication widening (4) of Section 3. The factor $k_{\ell-1}^{-1}$ is the normalization that makes the row sums equal to one.

We thus have nested families of widenings, all function-preserving: the block-wise widenings (6), the Kronecker widenings inside them, and the parallel widening and the duplication widening (4) inside those.

A.2 Widening by zero-padding neurons

The other natural way to view a narrow network as a wide one is to add neurons that are switched off rather than copies of existing ones. We record the equivalence this produces for future reference, followed by an explanation of why we do not use it as a building principle for our transferable GMNs.

Take $N_\ell \geq n_\ell$ arbitrary, put $m_\ell := N_\ell - n_\ell$ (so $m_0 = m_{L+1} = 0$), decompose $\mathbb{R}^{N_\ell} = \mathbb{R}^{n_\ell} \oplus \mathbb{R}^{m_\ell}$, and let

$$\varphi_\ell \colon \mathbb{R}^{n_\ell} \to \mathbb{R}^{N_\ell}, \qquad x \mapsto (x, 0). \tag{10}$$

The bias condition of (8) reads $b^{\uparrow(\ell)} = (b^{(\ell)}, 0)$. The activation condition now has real content: $\sigma(\varphi_\ell x) = (\sigma(x), \sigma(0)\mathbf{1}_{m_\ell})$, which equals $\varphi_\ell\sigma(x)$ for every x if and only if $\sigma(0) = 0$. Zero-padding therefore already restricts the architectures it applies to, holding for ReLU and $\tanh$ but failing for the sigmoid or any activation with a nonzero offset, whereas duplication holds for every elementwise activation.

Proposition 8 (The widenings compatible with zero-padding)**.** *Let φ_ℓ be as in* (10) *and write*

$$W^{\uparrow(\ell)} = \begin{pmatrix} {W^{\uparrow}}_{11}^{(\ell)} & {W^{\uparrow}}_{12}^{(\ell)} \\ {W^{\uparrow}}_{21}^{(\ell)} & {W^{\uparrow}}_{22}^{(\ell)} \end{pmatrix}$$

with the decompositions $\mathbb{R}^{N_{\ell-1}} = \mathbb{R}^{n_{\ell-1}} \oplus \mathbb{R}^{m_{\ell-1}}$ and $\mathbb{R}^{N_\ell} = \mathbb{R}^{n_\ell} \oplus \mathbb{R}^{m_\ell}$. Then for each ℓ, $W^{\uparrow(\ell)}\varphi_{\ell-1} = \varphi_\ell W^{(\ell)}$ if and only if ${W^{\uparrow}}_{11}^{(\ell)} = W^{(\ell)}$ and ${W^{\uparrow}}_{21}^{(\ell)} = 0$; the blocks ${W^{\uparrow}}_{12}^{(\ell)}$ and ${W^{\uparrow}}_{22}^{(\ell)}$ are unconstrained.

Proof. For the $\Leftarrow$ direction, block matrix multiplication gives $W^{\uparrow(\ell)}(x, 0) = \left({W^{\uparrow}}_{11}^{(\ell)}x, {W^{\uparrow}}_{21}^{(\ell)}x\right) = \left(W^{(\ell)}x, 0\right) = \varphi_\ell W^{(\ell)}x$ for every $x \in \mathbb{R}^{n_{\ell-1}}$.

For the $\Rightarrow$ direction, evaluate the identity at $x = e_j \in \mathbb{R}^{n_{\ell-1}}$ and read off the first n_ℓ and the last m_ℓ output coordinates separately, which gives ${W^{\uparrow}}_{11}^{(\ell)}e_j = W^{(\ell)}e_j$ and ${W^{\uparrow}}_{21}^{(\ell)}e_j = 0$. As j ranges over a basis of $\mathbb{R}^{n_{\ell-1}}$, this yields ${W^{\uparrow}}_{11}^{(\ell)} = W^{(\ell)}$ and ${W^{\uparrow}}_{21}^{(\ell)} = 0$. □

The wide weights compatible with zero-padding are therefore exactly

$$W^{\uparrow(\ell)} = \begin{pmatrix} W^{(\ell)} & * \\ 0 & * \end{pmatrix}, \qquad b^{\uparrow(\ell)} = \begin{pmatrix} b^{(\ell)} \\ 0 \end{pmatrix},$$

with the starred blocks arbitrary, provided $\sigma(0) = 0$. The pattern of constraints is exactly the statement that the padded neurons are dead. The zero blocks in the first column of $W^{\uparrow(\ell)}$ and the second block of $b^{\uparrow(\ell)}$ ensure that a padded neuron receives nothing, so its preactivation is 0 and, since $\sigma(0) = 0$, its activation is 0 as well; the starred blocks, which are the weights leaving the padded neurons, are then unconstrained because they are always applied to zeros.

Why we do not build on this structure. Zero-padding is arguably the more familiar of the two structures: it formalizes the intuition that a narrow network is a wide network with unused neurons. However, we argue that it is not the right structure to focus on for metanetworks, since we ultimately care about weights of different widths trained independently from random initialization (Sections 4 and 5). In a trained network of width N_ℓ, every hidden neuron is doing work: the initialization is exchangeable across the neurons of a layer, and training under a permutation-equivariant loss preserves that exchangeability, so no neuron is singled out as one that ought to be zero. In other words, there is a distributional permutation symmetry on weights trained from random initialization, so the wider network actually seen at test time, which we take to be a wide network trained from random initialization, is not expected to be close to a zero-padded version of a narrow network. Hence we do not expect invariance to zero-padding widening to be as useful for size generalization in practice as invariance to duplication widening. One might still ask whether imposing both invariances simultaneously could further improve size generalization. We discuss the obstruction to imposing both in Appendix B.5. Hence we must choose one invariance over the other.

A.3 The weight space as a consistent sequence

We now focus on the structure of the weight space arising from duplication of neurons (Appendix A.1). We refer to Appendix B of Levin et al. (2026) for the theoretical background that we follow here. Out of the nested families of widening operations mentioned there, only the duplication widening (4) fits the consistent-sequence framework of Levin et al. (2026) (reviewed briefly in Section 2.2). This is because a consistent sequence requires one designated linear embedding $V_n \hookrightarrow V_N$ per comparable pair of widths, whereas the block-wise family (6) singles out no map, and the Kronecker widening requires one arbitrary choice of $P^{(\ell)}$ per comparable pair of widths. This framework buys the two things the main text uses: a single space V_∞ holding weights of all widths at once, in which "the same function at a different width" becomes "the same point", and a distance on that space in which weights of different widths can be compared and continuity across widths can be stated.

Spaces, index set, and group action. For $n = (n_1, \dots, n_L)$, with $n_0 = d_{\text{in}}$ and $n_{L+1} = d_{\text{out}}$ held fixed, let V_n be the weight space of Section 3,

$$V_n := \bigoplus_{\ell=1}^{L+1} \left(\mathbb{R}^{n_\ell \times n_{\ell-1}} \oplus \mathbb{R}^{n_\ell} \right).$$

Order widths by divisibility, writing $n \preceq N$ when $n_i \mid N_i$ for every i. This makes the index set directed, since n and m are both refined by $N_i = \operatorname{lcm}(n_i, m_i)$, which is what the direct limit below needs.

Each V_n carries the hidden-neuron permutation symmetry. Let $\mathsf{G}_n := S_{n_1} \times \cdots \times S_{n_L}$, identify each permutation with its permutation matrix, and for $g = (\sigma_1, \dots, \sigma_L) \in \mathsf{G}_n$ set $\sigma_0 = I_{n_0}$ and $\sigma_{L+1} = I_{n_{L+1}}$, since input and output coordinates are fixed. The action of G_n on $\theta = \big((W^{(\ell)}, b^{(\ell)})\big)_{\ell=1}^{L+1} \in V_n$ is the familiar one,

$$(g \cdot \theta)^{(\ell)} := \left(\sigma_\ell W^{(\ell)} \sigma_{\ell-1}^{-1},\ \sigma_\ell b^{(\ell)} \right), \qquad \ell = 1, \dots, L+1,$$

which relabels the hidden activations by $x^{(\ell)} \mapsto \sigma_\ell x^{(\ell)}$ and leaves the represented function $\mathcal{R}(\theta)$ unchanged.

Embeddings. For $n \preceq N$ put $k_i = N_i / n_i$, with $k_0 = k_{L+1} = 1$, and let

$$\varphi_{N,n} \colon V_n \hookrightarrow V_N, \qquad \theta \mapsto \theta^{\uparrow}$$

be the duplication widening (4). In the rescaled coordinates the embedding is entrywise duplication,

$$\overline{W^{\uparrow}}^{(\ell)} = \overline{W}^{(\ell)} \otimes \mathbf{1}_{k_\ell} \mathbf{1}_{k_{\ell-1}}^{\top}, \qquad b^{\uparrow(\ell)} = b^{(\ell)} \otimes \mathbf{1}_{k_\ell}, \tag{11}$$

because $\overline{W^{\uparrow}}^{(\ell)} = N_{\ell-1} W^{\uparrow(\ell)} = k_{\ell-1} n_{\ell-1} \cdot k_{\ell-1}^{-1} \left(W^{(\ell)} \otimes \mathbf{1}_{k_\ell} \mathbf{1}_{k_{\ell-1}}^{\top} \right)$. This is the form we use below, and it is also why the first modification of the duplication-compatible GMN in Section 3

rescales the edge features by the fan-in: in these coordinates a widening does nothing but repeat features.

Furthermore, set the embedding of groups

$$\mathsf{G}_n \hookrightarrow \mathsf{G}_N, \qquad g \mapsto g^{\uparrow} := (\sigma_1 \otimes I_{k_1}, \dots, \sigma_L \otimes I_{k_L})$$

.

The transition maps compose under successive duplication widenings, and the group embeddings are equivariant with respect to those transition maps. Thus $(V_n, \mathsf{G}_n, \varphi_{N,n})$ satisfies Definition B.1 of Levin et al. (2026), so it is a consistent sequence.

In the rescaled coordinates (11), this consistent sequence can be seen as a direct sum of the duplication consistent sequence on sets in Appendix E.1 of Levin et al. (2026), except that the group actions are shared across layers.

We then produce the limit space (following Definitions B.2 and B.3 of Levin et al. (2026))

$$V_\infty := \varinjlim_n V_n, \qquad \mathsf{G}_\infty := \varinjlim_n \mathsf{G}_n.$$

Concretely, one may picture V_∞ as $\bigcup_n V_n$ with the identifications $\theta \sim \theta^{\uparrow}$ for every $n \preceq N$, identifying weights of different widths that are related via the duplication widening. The group G_∞ acts on V_∞ by permuting the hidden neurons of each width.

Compatible norms. We now equip V_∞ with a norm that compares weights of different widths, following Appendix B.3 of Levin et al. (2026). We use three norms on V_n: the *normalized spectral norm*

$$\|\theta\|_{V_n,\mathrm{op}} := \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|}{\sqrt{n_\ell n_{\ell-1}}} + \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \right) = \sum_{\ell=1}^{L+1} \left(\sqrt{\frac{n_{\ell-1}}{n_\ell}} \left\|W^{(\ell)}\right\| + \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \right), \tag{12}$$

the *normalized ℓ_1 norm*

$$\|\theta\|_{V_n,1} := \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|_1}{n_\ell n_{\ell-1}} + \frac{\left\|b^{(\ell)}\right\|_1}{n_\ell} \right) = \sum_{\ell=1}^{L+1} \left(\frac{\left\|W^{(\ell)}\right\|_1}{n_\ell} + \frac{\left\|b^{(\ell)}\right\|_1}{n_\ell} \right), \tag{13}$$

and, as an intermediate analytical tool used only in Appendix E.2, the *normalized Frobenius norm*

$$\|\theta\|^2_{V_n,\mathrm{F}} := \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|_{\mathrm{F}}^2}{n_\ell n_{\ell-1}} + \frac{\left\|b^{(\ell)}\right\|_2^2}{n_\ell} \right) = \sum_{\ell=1}^{L+1} \left(\frac{n_{\ell-1}}{n_\ell} \left\|W^{(\ell)}\right\|_{\mathrm{F}}^2 + \frac{\left\|b^{(\ell)}\right\|_2^2}{n_\ell} \right), \tag{14}$$

where $\|\cdot\|$ is the spectral norm, $\|\cdot\|_1$ the entrywise ℓ_1 norm, and $\|\cdot\|_{\mathrm{F}}$ the entrywise Frobenius norm. The two norms of (5) in the main text are exactly (12) and (13). Note that the normalized spectral and ℓ_1 norms defined here are weight-space analogues of the norms attached to duplication sequences on graphs in Appendix F.1 of Levin et al. (2026); the normalized Frobenius norm has no such role and is introduced purely as a technical device.

All three norms are G_n-invariant, and are preserved by the embeddings, which is precisely the reason for our choices of normalization factors in (12), (13), and (14). Concretely, for the spectral norm, $\|A \otimes B\| = \|A\|\|B\|$, and $\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top$ has rank one with $\left\|\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top\right\| = \|\mathbf{1}_{k_\ell}\|_2\|\mathbf{1}_{k_{\ell-1}}\|_2 = \sqrt{k_\ell k_{\ell-1}}$, so by (11)

$$\frac{\left\|\overline{W^{\uparrow}}^{(\ell)}\right\|}{\sqrt{N_\ell N_{\ell-1}}} = \frac{\left\|\overline{W}^{(\ell)}\right\|\sqrt{k_\ell k_{\ell-1}}}{\sqrt{k_\ell n_\ell\, k_{\ell-1} n_{\ell-1}}} = \frac{\left\|\overline{W}^{(\ell)}\right\|}{\sqrt{n_\ell n_{\ell-1}}}, \qquad \frac{\left\|b^{\uparrow(\ell)}\right\|_2}{\sqrt{N_\ell}} = \frac{\|b^{(\ell)}\|_2\sqrt{k_\ell}}{\sqrt{k_\ell n_\ell}} = \frac{\|b^{(\ell)}\|_2}{\sqrt{n_\ell}}.$$

For the ℓ_1 norm, $\|A \otimes B\|_1 = \|A\|_1 \|B\|_1$ and $\left\|\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top\right\|_1 = k_\ell k_{\ell-1}$, so

$$\frac{\left\|\overline{W^{\uparrow}}^{(\ell)}\right\|_1}{N_\ell N_{\ell-1}} = \frac{\left\|\overline{W}^{(\ell)}\right\|_1 k_\ell k_{\ell-1}}{k_\ell n_\ell\, k_{\ell-1} n_{\ell-1}} = \frac{\left\|\overline{W}^{(\ell)}\right\|_1}{n_\ell n_{\ell-1}}, \qquad \frac{\left\|b^{\uparrow(\ell)}\right\|_1}{N_\ell} = \frac{\left\|b^{(\ell)}\right\|_1 k_\ell}{k_\ell n_\ell} = \frac{\left\|b^{(\ell)}\right\|_1}{n_\ell}.$$

For the Frobenius norm, the same identity holds, $\|A \otimes B\|_{\mathrm{F}} = \|A\|_{\mathrm{F}} \|B\|_{\mathrm{F}}$, and $\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top$ again has $\left\|\mathbf{1}_{k_\ell}\mathbf{1}_{k_{\ell-1}}^\top\right\|_{\mathrm{F}} = \sqrt{k_\ell k_{\ell-1}}$, being rank one, so exactly as in the spectral-norm computation above,

$$\frac{\left\|\overline{W^{\uparrow}}^{(\ell)}\right\|_{\mathrm{F}}^2}{N_\ell N_{\ell-1}} = \frac{\left\|\overline{W}^{(\ell)}\right\|_{\mathrm{F}}^2}{n_\ell n_{\ell-1}}, \qquad \frac{\left\|b^{\uparrow(\ell)}\right\|_2^2}{N_\ell} = \frac{\left\|b^{(\ell)}\right\|_2^2}{n_\ell}.$$

Summing over ℓ gives $\|\varphi_{N,n}(\theta)\|_{V_N} = \|\theta\|_{V_n}$ for all three norms, so the embeddings are isometries, as claimed in Section 3. Therefore, all three norms are compatible with respect to the consistent sequence $(V_n, \mathsf{G}_n, \varphi_{N,n})$, in the sense of Definition 2.5 of Levin et al. (2026), and hence, they induce norms $\|\cdot\|_{V_\infty,\mathrm{op}}$, $\|\cdot\|_{V_\infty,1}$, or $\|\cdot\|_{V_\infty,\mathrm{F}}$ on V_∞, whose value at a point may be computed at any representative width (see Proposition B.8 of Levin et al. (2026)).

The symmetrized distances across widths. We further follow Definition B.9 of Levin et al. (2026) to define a symmetrized distance on the limit space that takes into account the hidden-neuron permutation symmetry. Let $\overline{V_\infty}$ denote the completion of V_∞ with respect to one of these norms; the G_∞-action extends from V_∞ to $\overline{V_\infty}$ by isometries. The norm already compares weights of different widths, but it insists on a particular labeling of the hidden neurons, which a metanetwork should not; we therefore symmetrize over G_∞,

$$d_\infty(x, y) := \inf_{g \in \mathsf{G}_\infty} \|g \cdot x - y\|_{\overline{V_\infty}}, \qquad x, y \in \overline{V_\infty}.$$

By Proposition B.10 of Levin et al. (2026), d_∞ is a pseudometric on $\overline{V_\infty}$ and a genuine metric on the set of orbit closures under the G_∞-action.

Different interpretations behind different choices of norm. We consider here two different norms on the weight space for the metanetwork itself, the normalized spectral norm (12) and the normalized ℓ_1 norm (13). The choice of operator norm is natural from the perspective of the represented function, since in an MLP the weight matrices act on the activations of the previous layer via matrix multiplication, and the operator norm is the natural way to measure how much a linear map can stretch a vector. The ℓ_1 norm is defined via entrywise inspection of the weight matrices, and hence it is natural from the perspective of a duplication-compatible GMN, which reads each weight entry through its own learnable map.

We will not discuss the interpretation of the normalized Frobenius norm (14) as it serves only as an analytical intermediary in Appendix E.2.

Comparing the three normalized norms. We now establish the relationship between the three norms in the limit space V_∞.

Proposition 9. *Both the normalized spectral norm and the normalized ℓ_1 norm are dominated by the normalized Frobenius norm: for every $\theta \in V_\infty$,*

$$\max\{\|\theta\|_{V_\infty,\mathrm{op}}, \|\theta\|_{V_\infty,1}\} \le \sqrt{2(L+1)}\, \|\theta\|_{V_\infty,\mathrm{F}}.$$

And consequently, the symmetrized distances also satisfy the same domination inequalities: for every $x, y \in \overline{V_\infty}$,

$$\max\{d_\infty^{\mathrm{op}}(x, y), d_\infty^{1}(x, y)\} \le \sqrt{2(L+1)}\, d_\infty^{\mathrm{F}}(x, y).$$

Proof. For each width n and every $\theta \in V_n$,

$$\|\theta\|_{V_n,\mathrm{op}} = \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|}{\sqrt{n_\ell n_{\ell-1}}} + \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \right)$$

$$\leq \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|_{\mathrm{F}}}{\sqrt{n_\ell n_{\ell-1}}} + \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \right) \qquad (\|A\| \leq \|A\|_{\mathrm{F}})$$

$$\leq \sqrt{2(L+1)}\|\theta\|_{V_n,\mathrm{F}} \qquad \text{(Cauchy–Schwarz on } 2(L+1) \text{ terms)}$$

Similarly,

$$\|\theta\|_{V_n,1} = \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|_1}{n_\ell n_{\ell-1}} + \frac{\left\|b^{(\ell)}\right\|_1}{n_\ell} \right)$$

$$\leq \sum_{\ell=1}^{L+1} \left(\frac{\left\|\overline{W}^{(\ell)}\right\|_{\mathrm{F}}}{\sqrt{n_\ell n_{\ell-1}}} + \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \right) \qquad \text{(Cauchy–Schwarz on matrix and vector entries)}$$

$$\leq \sqrt{2(L+1)}\|\theta\|_{V_n,\mathrm{F}} \qquad \text{(Cauchy–Schwarz on } 2(L+1) \text{ terms)}$$

Since the embeddings $\varphi_{N,n}$ are isometries for all three norms, the same inequalities hold on V_∞.

For the symmetrized distances, for any $g \in \mathsf{G}_\infty$,

$$\|g \cdot x - y\|_{V_\infty,\mathrm{op}} \leq \sqrt{2(L+1)}\|g \cdot x - y\|_{V_\infty,\mathrm{F}}.$$

Taking the infimum over $g \in \mathsf{G}_\infty$ gives $d_\infty^{\mathrm{op}}(x,y) \leq \sqrt{2(L+1)}d_\infty^{\mathrm{F}}(x,y)$, and similarly for d_∞^1. □

Proposition 10. *In the case of 1-layer MLP, $L = 1$, the normalized Frobenius norm and the normalized ℓ_1 norm are dominated by the normalized spectral norm:*

$$\max\{\|\theta\|_{V_\infty,\mathrm{F}}, \|\theta\|_{V_\infty,1}\} \leq \max\{1, \sqrt{d_{\mathrm{in}}}, \sqrt{d_{\mathrm{out}}}\}\, \|\theta\|_{V_\infty,\mathrm{op}}.$$

Proof. For each width n sufficiently large and every $\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}) \in V_n$, the two matrix blocks $\overline{W}^{(1)}$ and $\overline{W}^{(2)}$ have rank at most d_{in} and at most d_{out}, respectively, so the rank inequality $\|A\|_{\mathrm{F}} \leq \sqrt{\mathrm{rank}(A)}\,\|A\|$ gives

$$\left\|\overline{W}^{(1)}\right\|_{\mathrm{F}} \leq \sqrt{d_{\mathrm{in}}}\left\|\overline{W}^{(1)}\right\|, \qquad \left\|\overline{W}^{(2)}\right\|_{\mathrm{F}} \leq \sqrt{d_{\mathrm{out}}}\left\|\overline{W}^{(2)}\right\|.$$

Then,

$$\|\theta\|_{V_n,\mathrm{F}} \leq \frac{\left\|\overline{W}^{(1)}\right\|_{\mathrm{F}}}{\sqrt{nd_{\mathrm{in}}}} + \frac{\left\|b^{(1)}\right\|_2}{\sqrt{n}} + \frac{\left\|\overline{W}^{(2)}\right\|_{\mathrm{F}}}{\sqrt{nd_{\mathrm{out}}}} + \frac{\left\|b^{(2)}\right\|_2}{\sqrt{d_{\mathrm{out}}}} \qquad (\textstyle\sum x_i^2 \leq (\sum x_i)^2 \text{ for } x_i \geq 0)$$

$$\leq \frac{\sqrt{d_{\mathrm{in}}}\left\|\overline{W}^{(1)}\right\|}{\sqrt{nd_{\mathrm{in}}}} + \frac{\left\|b^{(1)}\right\|_2}{\sqrt{n}} + \frac{\sqrt{d_{\mathrm{out}}}\left\|\overline{W}^{(2)}\right\|}{\sqrt{nd_{\mathrm{out}}}} + \frac{\left\|b^{(2)}\right\|_2}{\sqrt{d_{\mathrm{out}}}} \qquad \text{(rank inequality above)}$$

$$\leq \max\{1, \sqrt{d_{\mathrm{in}}}, \sqrt{d_{\mathrm{out}}}\}\, \|\theta\|_{V_n,\mathrm{op}}.$$

Similarly,

$$\|\theta\|_{V_n,1} = \frac{\left\|\overline{W}^{(1)}\right\|_1}{nd_{\mathrm{in}}} + \frac{\left\|b^{(1)}\right\|_1}{n} + \frac{\left\|\overline{W}^{(2)}\right\|_1}{nd_{\mathrm{out}}} + \frac{\left\|b^{(2)}\right\|_1}{d_{\mathrm{out}}}$$

$$\leq \frac{\left\|\overline{W}^{(1)}\right\|_{\mathrm{F}}}{\sqrt{nd_{\mathrm{in}}}} + \frac{\left\|b^{(1)}\right\|_2}{\sqrt{n}} + \frac{\left\|\overline{W}^{(2)}\right\|_{\mathrm{F}}}{\sqrt{nd_{\mathrm{out}}}} + \frac{\left\|b^{(2)}\right\|_2}{\sqrt{d_{\mathrm{out}}}}$$

(Cauchy–Schwarz on matrix and vector entries)

$$\leq \max\{1, \sqrt{d_{\mathrm{in}}}, \sqrt{d_{\mathrm{out}}}\}\, \|\theta\|_{V_n,\mathrm{op}}.$$

Since the embeddings $\varphi_{N,n}$ are isometries for all three norms, the same inequalities hold on V_∞. □

In all depths, neither the normalized spectral norm nor the normalized Frobenius norm is dominated by the normalized ℓ_1 norm with a width-independent constant. When $L \geq 2$, neither the normalized Frobenius norm nor the normalized ℓ_1 norm is dominated by the normalized spectral norm with a width-independent constant.

B THEORETICAL PROPERTIES OF TRANSFERABLE GMNS

In Section 3, we propose three modifications to the GMN to make it duplication-compatible: fan-in rescaling of the input networks, mean aggregation, and a layer-wise mean readout. We then discuss its two shortcomings, and propose the matrix-product GMN to address them. We provide all the proofs of the theoretical properties of the duplication-compatible GMN and the matrix-product GMN in this section of Appendix.

Appendix B.1 proves the duplication-compatible GMN's duplication compatibility. Appendix B.2 shows that this comes at the cost of incompatibility with the larger family of block-wise widenings, and that the matrix-product GMN repairs this. Appendix B.3 discusses a further shortcoming of the duplication-compatible GMN, task-model misalignment. Appendix B.4 proves the matrix-product GMN's Lipschitz continuity with respect to the normalized spectral norm. Finally, Appendix B.5 shows that our modified GMNs cannot be further made compatible with zero-padding widening discussed in Appendix A.2.

B.1 DUPLICATION-COMPATIBILITY

We prove duplication compatibility for the simple GMN of Section 2.1 after the three modifications of Section 3. The point of the proof is to show that, at each round of message passing, the node and edge features on the wide network's computation graph are literally copies of the ones on the narrow network's computation graph.

Proof of Proposition 1. Let $\theta^{\uparrow}$ be the duplication widening (4) of θ with width multipliers $k_1, \dots, k_L$ and $k_0 = k_{L+1} = 1$, and write $v_{i,a}^{(\ell)}$, $a \in [k_\ell]$, for the copies of the narrow node $v_i^{(\ell)}$. We prove by induction on the message-passing round $t = 0, 1, \dots, T$ that

$$h_{i,a}^{(\ell),[t]} = h_i^{(\ell),[t]}, \qquad e_{(j,b)(i,a)}^{(\ell),[t]} = e_{ji}^{(\ell),[t]}$$

for every layer ℓ, every node index i and predecessor index j, and every pair of copy indices a, b.

Base case ($t = 0$). For the duplicated edge from $v_{j,b}^{(\ell-1)}$ to $v_{i,a}^{(\ell)}$, fan-in rescaling gives

$$\overline{W^{\uparrow}}_{(i,a)(j,b)}^{(\ell)} = n_{\ell-1} k_{\ell-1} \cdot k_{\ell-1}^{-1} W_{ij}^{(\ell)} = n_{\ell-1} W_{ij}^{(\ell)} = \overline{W}_{ij}^{(\ell)},$$

so every edge copy carries the same raw edge feature as the narrow edge, and the bias part of (4) gives $b_{i,a}^{(\ell)\uparrow} = b_i^{(\ell)}$, so every node copy carries the same raw node feature. The initial node and edge maps ψ_v, ψ_e of the simple GMN are shared across nodes and edges of the same type, so applying them to these identical raw features gives

$$h_{i,a}^{(\ell),[0]} = h_i^{(\ell),[0]}, \qquad e_{(j,b)(i,a)}^{(\ell),[0]} = e_{ji}^{(\ell),[0]},$$

which is the claim at $t = 0$.

Inductive step. Assume the claim holds at round t. Fixing a base predecessor j, the shared message map evaluated on identical edge and source-node features gives, for every copy,

$$m_{(j,b)\to(i,a)}^{(\ell),[t]} = \psi_m\left(e_{(j,b)(i,a)}^{(\ell),[t]}, h_{j,b}^{(\ell-1),[t]}\right) = \psi_m\left(e_{ji}^{(\ell),[t]}, h_j^{(\ell-1),[t]}\right) = m_{j\to i}^{(\ell),[t]},$$

so the forward mean aggregate at the copied node is

$$\frac{1}{n_{\ell-1} k_{\ell-1}} \sum_{j \in [n_{\ell-1}]} \sum_{b \in [k_{\ell-1}]} m_{(j,b)\to(i,a)}^{(\ell),[t]} = \frac{1}{n_{\ell-1}} \sum_{j \in [n_{\ell-1}]} m_{j\to i}^{(\ell),[t]},$$

with the same calculation giving the backward aggregate in the bidirectional GMN, with $n_{\ell+1} k_{\ell+1}$ in place of $n_{\ell-1} k_{\ell-1}$. The copied and narrow node features are then updated by the same shared map ψ_h applied to the same arguments, hence remain equal at round $t + 1$; the edge update is likewise shared and depends only on the old edge feature and the two endpoint node features, which are equal by the induction hypothesis, so copied edge features also remain equal at round $t + 1$. This completes the induction.

Graph-level readout. Since ψ_r is applied pointwise and the readout pools by a per-layer mean, the hidden-layer summaries agree:

$$\frac{1}{n_\ell k_\ell}\sum_{i\in[n_\ell]}\sum_{a\in[k_\ell]}\psi_r\left(h_{i,a}^{(\ell),[T]}\right)=\frac{1}{n_\ell}\sum_{i\in[n_\ell]}\psi_r\left(h_i^{(\ell),[T]}\right).$$

Since $k_0=k_{L+1}=1$, the input and output widths are fixed, so their coordinatewise states are listed in the same order for θ and $\theta^\uparrow$; the final readout map ρ therefore receives the same input for both, and $f^\Theta(\theta^\uparrow)=f^\Theta(\theta)$. □

B.2 Compatibility with block-wise widening

Incompatibility of the duplication-compatible GMN. We show the incompatibility of the duplication-compatible GMN with the full block-wise widening family (6) by constructing a specific counterexample.

The narrow MLP. Let θ be a scalar-input, scalar-output MLP with one hidden layer and no biases, with weights

$$W^{(1)}=w,\qquad W^{(2)}=1,$$

so it represents the function $x\mapsto\sigma(wx)$ for activation function σ.

The widened MLP. Let $\theta^\uparrow$ widen only the hidden layer, to width 2, with weights

$$W^{\uparrow(1)}=\begin{bmatrix}w\\w\end{bmatrix},\qquad W^{\uparrow(2)}=\begin{bmatrix}1 & 0\end{bmatrix}.$$

This satisfies the block-wise widening condition (6) but not the duplication widening (4), which would instead split the hidden-to-output weight evenly as $(\frac{1}{2},\frac{1}{2})$. It represents the same function as θ: both hidden copies compute $\sigma(wx)$, and the output recombines them as $1\cdot\sigma(wx)+0\cdot\sigma(wx)=\sigma(wx)$.

The GMN. Let the initial edge map be the identity, $\psi_e=\mathrm{id}$ and the initial node map ψ_v be arbitrary. On the narrower network θ, the output edge carries initial edge feature 1. On the widened network $\theta^\uparrow$, the two output edges carry features 2 and 0. Choose the message MLP $\psi_m(e,h)=\mathrm{ReLU}(e-\frac{1}{2})$, which discards the source-node feature h; choose the node-update MLP to copy the mean incoming message, $\psi_h(h,m^\rightarrow,m^\leftarrow)=m^\rightarrow$; and choose the final readout ρ to select the resulting output-node state.

The two outputs. On θ, the output-node state after one round is $\mathrm{ReLU}(1-\frac{1}{2})=\frac{1}{2}$. On $\theta^\uparrow$, mean aggregation over the $k_1=2$ incoming edges gives

$$\frac{1}{2}\,\mathrm{ReLU}\left(2-\tfrac{1}{2}\right)+\frac{1}{2}\,\mathrm{ReLU}\left(0-\tfrac{1}{2}\right)=\frac{3}{4}.$$

Since $\frac{1}{2}\neq\frac{3}{4}$, this GMN's two graph-level outputs differ although θ and $\theta^\uparrow$ represent the same function.

Compatibility of the matrix-product GMN.

Proof of Proposition 3(ii). **We first show that the forward matrix-product GMN is fully compatible with block-wise widenings** (6)**.**

Let $\theta^\uparrow$ be a block-wise widening (6) of θ with multipliers $k_1,\dots,k_L$ and $k_0=k_{L+1}=1$, so the widened widths are $N_\ell=n_\ell k_\ell$, and write $v_{i,a}^{(\ell)}$, $a\in[k_\ell]$, for the copies of the narrow node $v_i^{(\ell)}$. We prove by induction on $t=0,\dots,T$ that

$$h_{i,a}^{(\ell),[t]\uparrow}=h_i^{(\ell),[t]}\qquad\text{for every }\ell,\ i\in[n_\ell],\ a\in[k_\ell].\tag{15}$$

Base case ($t=0$). The bias part of (6) gives $b_{i,a}^{(\ell)\uparrow}=b_i^{(\ell)}$, so the shared initial node map ψ_v returns the same feature for every copy, and the input nodes carry the same constant at both widths.

Inductive step. Assume (15) at round t. Then,

$$
\begin{aligned}
z^{\mathrm{in},[t]\uparrow}_{v^{(\ell)}_{i,a}} &= \frac{1}{N_{\ell-1}} \sum_{j\in[n_{\ell-1}]} \sum_{b\in[k_{\ell-1}]} \overline{W^{\uparrow}}^{(\ell)}_{(i,a)(j,b)} A^{\mathrm{in}}_{\ell} h^{(\ell-1),[t]\uparrow}_{j,b} \\
&= \sum_{j\in[n_{\ell-1}]} \left(\sum_{b\in[k_{\ell-1}]} \left(W^{\uparrow(\ell)}_{ij}\right)_{ab} \right) A^{\mathrm{in}}_{\ell} h^{(\ell-1),[t]}_{j} \\
&= \sum_{j\in[n_{\ell-1}]} \left(W^{\uparrow(\ell)}_{ij} \mathbf{1}_{k_{\ell-1}}\right)_a A^{\mathrm{in}}_{\ell} h^{(\ell-1),[t]}_{j} = \sum_{j\in[n_{\ell-1}]} W^{(\ell)}_{ij} A^{\mathrm{in}}_{\ell} h^{(\ell-1),[t]}_{j} = z^{\mathrm{in},[t]}_{v^{(\ell)}_{i}},
\end{aligned}
$$

In the forward variant the node update therefore receives identical arguments at every copy, and since ψ_h is shared, (15) holds at round $t+1$, completing the induction.

Graph-level readout. Since ψ_r is applied pointwise and all k_ℓ copies in hidden layer ℓ carry the same feature by (15), the layer-wise mean collapses them back to the narrow summary, $\frac{1}{N_\ell}\sum_{i,a} \psi_r(h^{(\ell),[T]\uparrow}_{i,a}) = r_\ell$. The input and output node features are passed to ρ coordinatewise, in the same order at both widths because $k_0 = k_{L+1} = 1$. Hence $f^{\Theta}(\theta^{\uparrow}) = f^{\Theta}(\theta)$.

Next, we show that the bidirectional matrix-product GMN is compatible with the subclass of widenings that further satisfies the column-balance condition of Proposition 3(ii).

Inductive step, backward aggregate. For the bidirectional variant the same computation on the reversed edges gives

$$
z^{\mathrm{out},[t]\uparrow}_{v^{(\ell)}_{i,a}} = \frac{N_\ell}{N_{\ell+1}} \sum_{j\in[n_{\ell+1}]} \left(\mathbf{1}^{\top}_{k_{\ell+1}} W^{(\ell+1)\uparrow}_{ji}\right)_a A^{\mathrm{out}}_{\ell} h^{(\ell+1),[t]}_{j},
$$

whereas

$$
z^{\mathrm{out},[t]}_{v^{(\ell)}_{i}} = \frac{n_\ell}{n_{\ell+1}} \sum_{j\in[n_{\ell+1}]} W^{(\ell+1)}_{ji} A^{\mathrm{out}}_{\ell} h^{(\ell+1),[t]}_{j},
$$

because a copy of neuron i now collects the weights in the ath *column* of the block $W^{(\ell+1)\uparrow}_{ji}$. As $N_\ell/N_{\ell+1} = (k_\ell/k_{\ell+1})(n_\ell/n_{\ell+1})$, the column-balance condition

$$
\mathbf{1}^{\top}_{k_{\ell+1}} W^{(\ell+1)\uparrow}_{ji} = \frac{k_{\ell+1}}{k_\ell} W^{(\ell+1)}_{ji} \mathbf{1}^{\top}_{k_\ell} \qquad \text{for all } i, j,
$$

which is the condition of Proposition 3(ii) at layer $\ell+1$, makes the backward aggregate agree with the narrow one whenever (15) holds at round t. Together with the forward aggregate calculation above, the node update again receives the same arguments at every copy, so the induction and the readout argument go through as before. □

B.3 Task-model misalignment

Here we prove Proposition 2 regarding the misalignment between the target function and the duplication-compatible GMN model: they are continuous with respect to different norms on the weight space.

Continuity with respect to normalized spectral norm.

Proof of Proposition 2(i). **We first show the Lipschitz continuity of the realization map $\mathcal{R}$ with respect to the normalized spectral norm.** Since we have set up in Appendix A.3 a consistent sequence of weight spaces $(V_n, \mathsf{G}_n, \varphi_{N,n})$ with compatible norms $\|\cdot\|_{V_n,\mathrm{op}}$, it is enough, by Proposition C.2 of Levin et al. (2026), to show that in each finite space V_n, the realization map $\mathcal{R}$ is L_R Lipschitz on $\{\theta \in V_n : \|\theta\|_{V_n,\mathrm{op}} \le R\}$ with a constant independent of the hidden widths.

Let $\theta, \widetilde{\theta} \in V_n$ satisfy

$$
\|\theta\|_{V_n,\mathrm{op}} \le R, \qquad \left\|\widetilde{\theta}\right\|_{V_n,\mathrm{op}} \le R.
$$

Since the input space X is compact, choose R_X such that $\|u\|_2 \le R_X$ for every $u \in X$.

For $\eta \in V_n$ write $x_\eta^{(\ell)}(u)$ for the ℓ-th hidden-layer activation of the MLP with weights η on input $u \in X$, as given by (1). That is, $x_\eta^{(0)}(u) = u$ and $x_\eta^{(\ell)}(u) = \sigma\left(W_\eta^{(\ell)} x_\eta^{(\ell-1)}(u) + b_\eta^{(\ell)}\right)$ for $1 \le \ell \le L$. It is convenient to also write $x_\eta^{(L+1)}(u) := W_\eta^{(L+1)} x_\eta^{(L)}(u) + b_\eta^{(L+1)} = \mathcal{R}(\eta)(u)$ for the linear output layer, so that every $1 \le \ell \le L+1$ applies an affine map followed by a coordinatewise nonlinearity, that nonlinearity being the identity when $\ell = L+1$. Write $\mathcal{L}_\sigma$ for the Lipschitz constant of σ and set $\mathcal{L} := \max\{\mathcal{L}_\sigma, 1\}$, so that the nonlinearity of every layer $1 \le \ell \le L+1$ is $\mathcal{L}$-Lipschitz and maps $0 \in \mathbb{R}^{n_\ell}$ to a vector of norm at most $\sqrt{n_\ell}\,|\sigma(0)|$.

Define, for $0 \le \ell \le L+1$, the normalized activation size and discrepancy

$$\rho_\ell := \sup_{u \in X} \frac{\max\left\{\left\|x_\theta^{(\ell)}(u)\right\|_2, \left\|x_{\widetilde{\theta}}^{(\ell)}(u)\right\|_2\right\}}{\sqrt{n_\ell}}, \qquad d_\ell := \sup_{u \in X} \frac{\left\|x_\theta^{(\ell)}(u) - x_{\widetilde{\theta}}^{(\ell)}(u)\right\|_2}{\sqrt{n_\ell}},$$

and, for $1 \le \ell \le L+1$, the per-layer parameter discrepancy

$$\delta_\ell := \sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W_\theta^{(\ell)} - W_{\widetilde{\theta}}^{(\ell)}\right\| + \frac{\left\|b_\theta^{(\ell)} - b_{\widetilde{\theta}}^{(\ell)}\right\|_2}{\sqrt{n_\ell}}, \qquad \text{so that} \qquad \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}} = \sum_{\ell=1}^{L+1} \delta_\ell.$$

We prove by induction on the layer index ℓ that

$$\rho_\ell \le C_\ell, \qquad d_\ell \le C'_\ell \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}$$

for constants C_ℓ, C'_ℓ depending only on $R, R_X, \mathcal{L}_\sigma, \sigma(0), L, d_{\mathrm{in}}$ and not on the hidden widths. Throughout we use that each summand of $\|\eta\|_{V_n,\mathrm{op}}$ is bounded by the whole sum, so $\|\eta\|_{V_n,\mathrm{op}} \le R$ gives

$$\sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W_\eta^{(\ell)}\right\| \le R, \qquad \frac{\left\|b_\eta^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \le R, \qquad 1 \le \ell \le L+1, \tag{16}$$

and likewise $\delta_\ell \le \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}$.

Base case ($\ell = 0$). Both networks read the same input, $x_\theta^{(0)}(u) = x_{\widetilde{\theta}}^{(0)}(u) = u$, so the two claims hold with

$$C_0 := \frac{R_X}{\sqrt{d_{\mathrm{in}}}}, \qquad C'_0 := 0,$$

using $\|u\|_2 \le R_X$ for the first and $d_0 = 0$ for the second. Since $n_0 = d_{\mathrm{in}}$ is fixed, C_0 is independent of the hidden widths.

Induction step. Let $1 \le \ell \le L+1$ and assume the two bounds at $\ell - 1$. Write $z_\eta^{(\ell)}(u) := W_\eta^{(\ell)} x_\eta^{(\ell-1)}(u) + b_\eta^{(\ell)}$ for the pre-activation, so that $x_\eta^{(\ell)}(u)$ is the layer nonlinearity applied to $z_\eta^{(\ell)}(u)$. For the size bound, the triangle inequality, the bound $\left\|W_\eta^{(\ell)} v\right\|_2 \le \left\|W_\eta^{(\ell)}\right\| \|v\|_2$ defining the spectral norm, and (16) give, for $\eta \in \{\theta, \widetilde{\theta}\}$ and every $u \in X$,

$$\frac{\left\|z_\eta^{(\ell)}(u)\right\|_2}{\sqrt{n_\ell}} \le \sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W_\eta^{(\ell)}\right\| \frac{\left\|x_\eta^{(\ell-1)}(u)\right\|_2}{\sqrt{n_{\ell-1}}} + \frac{\left\|b_\eta^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \le R\left(\rho_{\ell-1} + 1\right).$$

Applying the nonlinearity, whose Lipschitz property and constant term contribute the factor $\mathcal{L}$ and the additive $|\sigma(0)|$ respectively, and then the induction hypothesis $\rho_{\ell-1} \le C_{\ell-1}$,

$$\rho_\ell \le \mathcal{L}\, R\left(\rho_{\ell-1} + 1\right) + |\sigma(0)| \le C_\ell := \mathcal{L}\, R\left(C_{\ell-1} + 1\right) + |\sigma(0)|.$$

For the discrepancy bound, split the difference of the two pre-activations into a weight part, a feature part, and a bias part:

$$z_\theta^{(\ell)}(u) - z_{\widetilde{\theta}}^{(\ell)}(u) = \left(W_\theta^{(\ell)} - W_{\widetilde{\theta}}^{(\ell)}\right) x_\theta^{(\ell-1)}(u) + W_{\widetilde{\theta}}^{(\ell)}\left(x_\theta^{(\ell-1)}(u) - x_{\widetilde{\theta}}^{(\ell-1)}(u)\right) + \left(b_\theta^{(\ell)} - b_{\widetilde{\theta}}^{(\ell)}\right).$$

Normalizing by $\sqrt{n_\ell}$ and estimating the three terms as above — the first two by the spectral-norm bound, the second also using (16) for $\widetilde{\theta}$ — the triangle inequality and the $\mathcal{L}$-Lipschitz property of the nonlinearity give

$$d_\ell \le \mathcal{L}\left(\rho_{\ell-1}\sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W_\theta^{(\ell)} - W_{\widetilde{\theta}}^{(\ell)}\right\| + R\,d_{\ell-1} + \frac{\left\|b_\theta^{(\ell)} - b_{\widetilde{\theta}}^{(\ell)}\right\|_2}{\sqrt{n_\ell}}\right)$$
$$\le \mathcal{L}\left((C_{\ell-1}+1)\,\delta_\ell + R\,d_{\ell-1}\right),$$

where the last step uses $\rho_{\ell-1} \le C_{\ell-1}$ on the weight term and $1 \le C_{\ell-1}+1$ on the bias term, which lets the two be collected into $(C_{\ell-1}+1)\delta_\ell$. By the induction hypothesis $d_{\ell-1} \le C'_{\ell-1}\left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}$ and by $\delta_\ell \le \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}$,

$$d_\ell \le C'_\ell\left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}, \qquad C'_\ell := \mathcal{L}\left(C_{\ell-1} + 1 + R\,C'_{\ell-1}\right).$$

Both recursions involve only $R, \mathcal{L}_\sigma, \sigma(0)$ and the constants at $\ell-1$, so C_ℓ and C'_ℓ are again independent of the hidden widths, completing the induction.

Conclusion. The case $\ell = L+1$ of the induction is the desired bound: since $x_\theta^{(L+1)}(u) = \mathcal{R}(\theta)(u)$ and $n_{L+1} = d_{\mathrm{out}}$ is fixed,

$$\left\|\mathcal{R}(\theta) - \mathcal{R}(\widetilde{\theta})\right\|_\infty = \sup_{u\in X}\left\|\mathcal{R}(\theta)(u) - \mathcal{R}(\widetilde{\theta})(u)\right\|_2 = \sqrt{d_{\mathrm{out}}}\,d_{L+1} \le L_R\left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}},$$

where $L_R := \sqrt{d_{\mathrm{out}}}\,C'_{L+1}$ is independent of the hidden widths.

We next show that, if σ is unbounded, the realization map need not be continuous with respect to the normalized ℓ_1 norm. Choose a sequence $a_m \in \mathbb{R}$ such that $|\sigma(a_m)| \to \infty$, and choose hidden widths n_m so that $n_m \ge m\,|a_m|$. Passing to a tail of the sequence if necessary, assume also that $\sigma(a_m) \ne 0$ for every m. Let $\theta_m = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ be the weights of a scalar-output, one-hidden-layer network with hidden width n_m, given by $W^{(1)} = 0 \in \mathbb{R}^{n_m\times d_{\mathrm{in}}}$, $b^{(1)} = a_m e_1 \in \mathbb{R}^{n_m}$, $W^{(2)} = \sigma(a_m)^{-1}e_1^\top \in \mathbb{R}^{1\times n_m}$, and $b^{(2)} = 0$, where e_1 is the first standard basis vector. Then

$$\|\theta_m\|_{V_{n_m},1} = \frac{|a_m|}{n_m} + \frac{1}{|\sigma(a_m)|} \le \frac{1}{m} + \frac{1}{|\sigma(a_m)|} \to 0.$$

However, for every $x \in X$,

$$\mathcal{R}(\theta_m)(x) = \sigma(a_m)^{-1}\sigma(a_m) = 1, \qquad \mathcal{R}(0)(x) = 0.$$

Hence $\theta_m \to 0$ in $\|\cdot\|_{V_\infty,1}$ but $\mathcal{R}(\theta_m)$ does not converge to $\mathcal{R}(0)$ in the uniform norm.

Consider the Lipschitz functional $F^*(g) = g(x_0)$, for any fixed $x_0 \in X$. Then the target function $F^*\circ\mathcal{R}$ is not continuous with respect to the normalized ℓ_1 norm, since $F^*\circ\mathcal{R}(\theta_m) = \mathcal{R}(\theta_m)(x_0) = 1$ for every m, while $F^*\circ\mathcal{R}(0) = \mathcal{R}(0)(x_0) = 0$, so $F^*\circ\mathcal{R}(\theta_m)$ does not converge to $F^*\circ\mathcal{R}(0)$ even though $\theta_m \to 0$ in $\|\cdot\|_{V_\infty,1}$. □

Continuity with respect to normalized ℓ_1 norm.

Proof of Proposition 2(ii). **We first show the Lipschitz continuity of the GMN map f^Θ with respect to the normalized ℓ_1 norm.** Fix the learnable GMN parameters Θ. Since we have set up in Appendix A.3 a consistent sequence of weight spaces $(V_n, \mathsf{G}_n, \varphi_{N,n})$ with compatible norms $\|\cdot\|_{V_n,1}$, it is enough, by Proposition C.2 of Levin et al. (2026), to show that in each finite space V_n, the GMN map f^Θ is Lipschitz with a constant independent of the hidden widths.

Let $\theta, \widetilde{\theta} \in V_n$, and for the node and edge features after round t define the average discrepancies

$$\Delta_t^{(\ell)} := \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left\|h_i^{(\ell),[t]} - \widetilde{h}_i^{(\ell),[t]}\right\|_1, \qquad \Gamma_t^{(\ell)} := \frac{1}{n_{\ell-1}n_\ell}\sum_{j=1}^{n_{\ell-1}}\sum_{i=1}^{n_\ell}\left\|e_{ji}^{(\ell),[t]} - \widetilde{e}_{ji}^{(\ell),[t]}\right\|_1,$$

with totals $\Delta_t := \sum_{\ell=0}^{L+1} \Delta_t^{(\ell)}$ and $\Gamma_t := \sum_{\ell=1}^{L+1} \Gamma_t^{(\ell)}$. Let $A_\Theta \geq 1$ be a common Lipschitz constant of the shared maps $\psi_v, \psi_e, \psi_m, \psi_m^{\leftarrow}, \psi_h, \psi_u$, a constant depending only on Θ.

We prove by induction on the message-passing round t that

$$\Delta_t + \Gamma_t \leq C \left\| \theta - \widetilde{\theta} \right\|_{V_n,1}$$

for constant C depending only on $\Theta, T, d_{\text{in}}, d_{\text{out}}$ and not on the hidden widths.

Base case ($t = 0$). At the start of the message-passing $t = 0$, we have $\Delta_0^{(0)} = 0$ and

$$\Delta_0^{(\ell)} \leq \frac{A_\Theta}{n_\ell} \left\| b^{(\ell)} - \widetilde{b}^{(\ell)} \right\|_1, \qquad \Gamma_0^{(\ell)} \leq \frac{A_\Theta}{n_{\ell-1} n_\ell} \left\| \overline{W}^{(\ell)} - \widetilde{\overline{W}}^{(\ell)} \right\|_1, \qquad \ell = 1, \ldots, L+1,$$

and so $\Delta_0 + \Gamma_0 \leq A_\Theta \left\| \theta - \widetilde{\theta} \right\|_{V_n,1}$.

Induction step. For $t \geq 0$, by the Lipschitz continuity of $\psi_m, \psi_m^{\leftarrow}, \psi_h, \psi_u$,

$$\Delta_{t+1}^{(\ell)} \leq A_\Theta^2 \left(\Delta_t^{(\ell)} + \Delta_t^{(\ell-1)} + \Delta_t^{(\ell+1)} + \Gamma_t^{(\ell)} + \Gamma_t^{(\ell+1)} \right),$$

$$\Gamma_{t+1}^{(\ell)} \leq A_\Theta \left(\Gamma_t^{(\ell)} + \Delta_t^{(\ell-1)} + \Delta_t^{(\ell)} \right),$$

with the conventions $\Delta_t^{(-1)} = \Delta_t^{(L+2)} = \Gamma_t^{(0)} = \Gamma_t^{(L+2)} = 0$ at the input and output layers. For the forward GMN, $\psi_m^{\leftarrow} = 0$ and hence the two terms $\Delta_t^{(\ell+1)}, \Gamma_t^{(\ell+1)}$ drop out.

Summing over ℓ, we have $\Delta_{t+1} \leq A_\Theta^2 (3\Delta_t + 2\Gamma_t)$ and $\Gamma_{t+1} \leq A_\Theta (2\Delta_t + \Gamma_t)$, so $\Delta_{t+1} + \Gamma_{t+1} \leq 5A_\Theta^2 (\Delta_t + \Gamma_t)$. Iterating for T rounds,

$$\Delta_T + \Gamma_T \leq \left(5A_\Theta^2\right)^T A_\Theta \left\| \theta - \widetilde{\theta} \right\|_{V_n,1}.$$

Graph-level readout. In the layer-wise mean readout, by the Lipschitz continuity of ψ_r and ρ,

$$\left\| f^\Theta(\theta) - f^\Theta(\widetilde{\theta}) \right\|_1 \leq C_{\Theta,T} \left\| \theta - \widetilde{\theta} \right\|_{V_n,1},$$

where $C_{\Theta,T}$ depends only on $\Theta, T, d_{\text{in}}, d_{\text{out}}$ and not on the hidden widths.

We now show that the duplication-compatible GMN class contains functions that are discontinuous in the normalized spectral norm. We construct a specific counterexample.

The weight sequence. It is enough to give one architecture for which discontinuity occurs, so take depth $L = 2$, scalar input and output ($n_0 = n_3 = 1$), and hidden widths $n_1 = n_2 = n$. Let n range over powers of two, and let $H_n \in \{\pm 1\}^{n \times n}$ be the Sylvester Hadamard matrix, so $H_n^\top H_n = nI_n$ and $\|H_n\| = \sqrt{n}$. Define weight $\theta_n \in V_n$ in the rescaled coordinates by $\overline{W}^{(2)} := H_n$ and all other rescaled weights and biases zero. Then

$$\|\theta_n\|_{V_n,\text{op}} = \frac{\|H_n\|}{\sqrt{n \cdot n}} = \frac{1}{\sqrt{n}} \xrightarrow[n \to \infty]{} 0,$$

hence $\theta_n \to 0$ in V_∞.

The GMN. Take the duplication-compatible GMN of Section 3 with $d_h = 1$, one round of message passing $T = 1$, forward messages only ($\psi_m^{\leftarrow} = 0$), and

$$\psi_v \equiv 0, \qquad \psi_e = \text{id}, \qquad \psi_m(e, h) = \text{ReLU}(e) + \text{ReLU}(-e) = |e|,$$

$$\psi_h(h, m^{\rightarrow}, m^{\leftarrow}) = m^{\rightarrow}, \qquad \psi_r = \text{id}, \qquad \rho\left(r_1, r_2, h_1^{(0),[T]}, h_1^{(3),[T]}\right) = r_2.$$

The edge update ψ_u may be taken to be zero, since with $T = 1$ its output is never read. All of these can be computed by ReLU-MLPs, so the GMN is realizable.

The function computed by this GMN is

$$f^\Theta(\theta) = \frac{1}{n_2} \sum_{i=1}^{n_2} \frac{1}{n_1} \sum_{j=1}^{n_1} \left| \overline{W}_{ij}^{(2)} \right|.$$

Every entry of H_n has modulus one, so $f^\Theta(\theta_n) = 1$ for every n while $f^\Theta(0) = 0$. Thus $\theta_n \to 0$ but $f^\Theta(\theta_n) \not\to f^\Theta(0)$. So the GMN map f^Θ is not continuous with respect to the normalized spectral norm. □

B.4 SPECTRAL CONTINUITY OF MATRIX-PRODUCT GMNS

Here we prove Proposition 3(i), the Lipschitz continuity of the matrix-product GMN with respect to the normalized spectral norm.

It is convenient to write the message passing in matrix form. Let $H^{(\ell),[t]} \in \mathbb{R}^{n_\ell \times d_h}$ be the matrix whose ith row is the node feature $h_i^{(\ell),[t]}$ after t rounds. Since the fan-in rescaling cancels the $1/n_{\ell-1}$ of the mean, the two mean aggregates of (7) are the rows of

$$Z^{(\ell),\mathrm{in},[t]} = W^{(\ell)} H^{(\ell-1),[t]} A_\ell^{\mathrm{in}}, \qquad Z^{(\ell),\mathrm{out},[t]} = \frac{n_\ell}{n_{\ell+1}} \left(W^{(\ell+1)}\right)^\top H^{(\ell+1),[t]} A_\ell^{\mathrm{out}}, \tag{17}$$

with the boundary conventions $Z^{(0),\mathrm{in},[t]} := 0$ and $Z^{(L+1),\mathrm{out},[t]} := 0$, and the node update applies the map ψ_h to the rows of $H^{(\ell),[t]}, Z^{(\ell),\mathrm{in},[t]}, Z^{(\ell),\mathrm{out},[t]}$. The forward variant is the case $A_\ell^{\mathrm{out}} = 0$. Note that, unlike the duplication-compatible GMN, this model carries no learned edge features: the weights enter only through (17).

Proof of Proposition 3(i). Since Appendix A.3 sets up the consistent sequence $(V_n, \mathsf{G}_n, \varphi_{N,n})$ with the compatible norms $\|\cdot\|_{V_n,\mathrm{op}}$, it is enough, by Proposition C.2 of Levin et al. (2026), to prove for each finite space V_n a Lipschitz bound on $\{\theta \in V_n : \|\theta\|_{V_n,\mathrm{op}} \le R\}$ with a constant independent of the hidden widths $n_1, \dots, n_L$.

Let $A_\Theta \ge 1$ be a common Lipschitz constant of the shared maps $\psi_v, \psi_h, \psi_r, \rho$, and let $a_\Theta \ge 1$ bound the spectral norms $\|A_\ell^{\mathrm{in}}\|, \|A_\ell^{\mathrm{out}}\|$ for all ℓ; both depend only on Θ. Fix $\theta, \widetilde{\theta} \in V_n$ with $\|\theta\|_{V_n,\mathrm{op}}, \left\|\widetilde{\theta}\right\|_{V_n,\mathrm{op}} \le R$, decorate with a tilde every quantity computed from $\widetilde{\theta}$, and set

$$M_t^{(\ell)} := \frac{\left\|H^{(\ell),[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}}, \quad \Delta_t^{(\ell)} := \frac{\left\|H^{(\ell),[t]} - \widetilde{H}^{(\ell),[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}}, \quad M_t := \sum_{\ell=0}^{L+1} M_t^{(\ell)}, \quad D_t := \sum_{\ell=0}^{L+1} \Delta_t^{(\ell)}.$$

We prove by induction on the message-passing round t that

$$M_t, \widetilde{M}_t \le K_{\Theta,t,R}, \qquad D_t \le C_{\Theta,t,R} \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}$$

for constants $C_{\Theta,t,R}, K_{\Theta,t,R}$ depending only on $\Theta, t, R, L, d_{\mathrm{in}}, d_{\mathrm{out}}$ and not on the hidden widths.

We use two elementary facts. First, the aggregates (17) obey

$$\frac{\left\|Z^{(\ell),\mathrm{in},[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}} \le a_\Theta \sqrt{\frac{n_{\ell-1}}{n_\ell}} \left\|W^{(\ell)}\right\| M_t^{(\ell-1)} \le a_\Theta R M_t^{(\ell-1)}, \qquad \frac{\left\|Z^{(\ell),\mathrm{out},[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}} \le a_\Theta R M_t^{(\ell+1)}. \tag{18}$$

Second, if a map ψ of q arguments with Lipschitz constant A is applied to the rows of matrices $X_1, \dots, X_q$ with m rows each, then

$$\begin{aligned} \|\psi(X_1, \dots, X_q) - \psi(Y_1, \dots, Y_q)\|_{\mathrm{F}} &\le \left(A^2 \sum_{j=1}^{q} \|X_j - Y_j\|_{\mathrm{F}}^2 \right)^{1/2} \le A \sum_{j=1}^{q} \|X_j - Y_j\|_{\mathrm{F}}, \\ \|\psi(X_1, \dots, X_q)\|_{\mathrm{F}} &\le \sqrt{m} \|\psi(0)\|_2 + A \sum_{j=1}^{q} \|X_j\|_{\mathrm{F}}. \end{aligned} \tag{19}$$

Base case ($t = 0$). The input nodes carry a constant, so $H^{(0),[0]} = \widetilde{H}^{(0),[0]}$ is a fixed matrix with d_{in} rows, and $h_i^{(\ell),[0]} = \psi_v(b_i^{(\ell)})$ for $\ell \ge 1$. Hence, by (19),

$$M_0^{(\ell)} \le \|\psi_v(0)\|_2 + A_\Theta \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \le \|\psi_v(0)\|_2 + A_\Theta R, \qquad \ell \ge 1,$$

and $M_0^{(0)}$ is a constant depending only on Θ and d_{in}, so $M_0, \widetilde{M}_0 \le K_{\Theta,R}$. Likewise $\Delta_0^{(0)} = 0$ and $\Delta_0^{(\ell)} \le A_\Theta \left\|b^{(\ell)} - \widetilde{b}^{(\ell)}\right\|_2 / \sqrt{n_\ell}$ for $\ell \ge 1$, whose sum over ℓ gives

$$D_0 \le A_\Theta \left\|\theta - \widetilde{\theta}\right\|_{V_n,\mathrm{op}}.$$

Induction step. Combining (19) with (18),

$$M_{t+1}^{(\ell)} \le \|\psi_h(0)\|_2 + A_\Theta \left(M_t^{(\ell)} + a_\Theta R\, M_t^{(\ell-1)} + a_\Theta R\, M_t^{(\ell+1)} \right),$$

with the conventions $M_t^{(-1)} = M_t^{(L+2)} = 0$. Summing over ℓ gives the scalar recurrence $M_{t+1} \le (L+2)\|\psi_h(0)\|_2 + A_\Theta(1 + 2a_\Theta R) M_t$. By induction, there exists a constant $K_{\Theta,t+1,R}$ depending only on $\Theta, t+1, R, L, d_{\text{in}}, d_{\text{out}}$ such that

$$M_{t+1}, \widetilde{M}_{t+1} \le K_{\Theta,t+1,R}.$$

Splitting the difference of two forward aggregates into a weight part and a feature part,

$$Z^{(\ell),\text{in},[t]} - \widetilde{Z}^{(\ell),\text{in},[t]} = \left(W^{(\ell)} - \widetilde{W}^{(\ell)} \right) H^{(\ell-1),[t]} A_\ell^{\text{in}} + \widetilde{W}^{(\ell)} \left(H^{(\ell-1),[t]} - \widetilde{H}^{(\ell-1),[t]} \right) A_\ell^{\text{in}},$$

and estimating each term as in (18), using the induction bound on M_t for the first and $\left\|\widetilde{\theta}\right\|_{V_n,\text{op}} \le R$ on the second, gives

$$\begin{aligned} \frac{\left\| Z^{(\ell),\text{in},[t]} - \widetilde{Z}^{(\ell),\text{in},[t]} \right\|_{\text{F}}}{\sqrt{n_\ell}} &\le a_\Theta \left(K_{\Theta,t,R} \sqrt{\tfrac{n_{\ell-1}}{n_\ell}} \left\| W^{(\ell)} - \widetilde{W}^{(\ell)} \right\| + R\, \Delta_t^{(\ell-1)} \right) \\ &\le a_\Theta \left(K_{\Theta,t,R} \left\| \theta - \widetilde{\theta} \right\|_{V_n,\text{op}} + R\, \Delta_t^{(\ell-1)} \right), \end{aligned}$$

The same computation on $(W^{(\ell+1)})^\top$ bounds the backward discrepancy by the same expression with $\Delta_t^{(\ell+1)}$ in place of $\Delta_t^{(\ell-1)}$. Feeding these into (19) for ψ_h and summing over ℓ,

$$D_{t+1} \le A_\Theta(1 + 2a_\Theta R) D_t + 2(L+2) A_\Theta a_\Theta K_{\Theta,t,R} \left\| \theta - \widetilde{\theta} \right\|_{V_n,\text{op}}.$$

By induction, $D_{t+1} \le C_{\Theta,t+1,R} \left\| \theta - \widetilde{\theta} \right\|_{V_n,\text{op}}$ for a constant $C_{\Theta,t+1,R}$ depending only on $\Theta, t+1, R, L, d_{\text{in}}, d_{\text{out}}$ and not on the hidden widths.

Graph-level readout. For a hidden layer, Cauchy–Schwarz applied to the layer-wise mean $r_\ell = \frac{1}{n_\ell} \sum_{i \in [n_\ell]} \psi_r(h_i^{(\ell),[T]})$ gives

$$\|r_\ell - \widetilde{r}_\ell\|_2 \le \frac{A_\Theta}{n_\ell} \sum_{i \in [n_\ell]} \left\| h_i^{(\ell),[T]} - \widetilde{h}_i^{(\ell),[T]} \right\|_2 \le \frac{A_\Theta}{\sqrt{n_\ell}} \left\| H^{(\ell),[T]} - \widetilde{H}^{(\ell),[T]} \right\|_{\text{F}} = A_\Theta \Delta_T^{(\ell)}.$$

The input and output node features enter ρ coordinatewise, and the Euclidean norm of their concatenated discrepancy is $\sqrt{d_{\text{in}}}\Delta_T^{(0)} + \sqrt{d_{\text{out}}}\Delta_T^{(L+1)}$ at most, with $d_{\text{in}}, d_{\text{out}}$ fixed. Since ρ is A_Θ-Lipschitz,

$$\left\| f^\Theta(\theta) - f^\Theta(\widetilde{\theta}) \right\|_2 \le C'_{\Theta,T,R} D_T \le C''_{\Theta,T,R} \left\| \theta - \widetilde{\theta} \right\|_{V_n,\text{op}},$$

which is the required width-uniform Lipschitz bound on V_n. □

B.5 Zero-padding compatibility

We discuss whether our proposed transferable GMNs are compatible with the zero-padding widening of Appendix A.2, and if not, whether they can be modified to be so.

Duplication-compatible GMNs. The duplication-compatible GMN cannot be made zero-padding compatible using a similar recipe. Indeed, suppose the widened weight is

$$W^{\uparrow(\ell)} = \begin{pmatrix} W^{(\ell)} & B^{(\ell)} \\ 0 & D^{(\ell)} \end{pmatrix}$$

with respect to $\mathbb{R}^{N_{\ell-1}} = \mathbb{R}^{n_{\ell-1}} \oplus \mathbb{R}^{m_{\ell-1}}$ and $\mathbb{R}^{N_\ell} = \mathbb{R}^{n_\ell} \oplus \mathbb{R}^{m_\ell}$, with $B^{(\ell)}$ and $D^{(\ell)}$ arbitrary (Appendix A.2). Suppose also that the source-node features are zero-padded: the original nodes

$v_j^{(\ell-1)}$, $j \in [n_{\ell-1}]$, keep the narrow feature $h_j^{(\ell-1),[t]}$, while the new nodes $v_a^{(\ell-1)}$, $a \in [m_{\ell-1}]$, carry $h_a^{(\ell-1),[t]} = 0$. For a forward edgewise message rule and a target node $v_i^{(\ell)}$ among the original ones, $i \in [n_\ell]$, the wide mean aggregate splits into a sum over the $n_{\ell-1}$ old edges and a sum over the $m_{\ell-1}$ new edges:

$$\frac{1}{N_{\ell-1}} \sum_{j=1}^{n_{\ell-1}} \psi_m \left(N_{\ell-1} W_{ij}^{(\ell)}, h_j^{(\ell-1),[t]} \right) + \frac{1}{N_{\ell-1}} \sum_{a=1}^{m_{\ell-1}} \psi_m \left(N_{\ell-1} B_{ia}^{(\ell)}, 0 \right),$$

whereas the narrow aggregate at the same node is

$$\frac{1}{n_{\ell-1}} \sum_{j=1}^{n_{\ell-1}} \psi_m \left(n_{\ell-1} W_{ij}^{(\ell)}, h_j^{(\ell-1),[t]} \right).$$

These two expressions need not agree in general.

Forward matrix-product GMNs, before readout. The forward matrix-product aggregation can be easily made compatible with the zero-padding widening of Appendix A.2 at the level of message passing (before final readout).

If the source-node features are zero-padded,

$$H^{(\ell-1),[t]\uparrow} = \begin{pmatrix} H^{(\ell-1),[t]} \\ 0 \end{pmatrix},$$

then (17) gives

$$Z^{(\ell),\mathrm{in},[t]\uparrow} = W^{\uparrow(\ell)} H^{(\ell-1),[t]\uparrow} A_\ell^{\mathrm{in}} = \begin{pmatrix} W^{(\ell)} H^{(\ell-1),[t]} A_\ell^{\mathrm{in}} \\ 0 \end{pmatrix},$$

so the original nodes receive the narrow aggregate and the padded ones receive zero. Propagating the zero-padded relation through the rest of the forward model would additionally require the preprocessing and the node update to preserve zeros, for example $\psi_v(0) = 0$ and $\psi_h(0, 0) = 0$ in the forward-only update.

Bidirectional matrix-product GMN, before readout. The bidirectional matrix-product aggregation is not compatible with the full zero-padding family in Appendix A.2. Indeed, if the successor-node features are zero-padded, then the backward part of (17) becomes

$$Z^{(\ell),\mathrm{out},[t]\uparrow} = \frac{N_\ell}{N_{\ell+1}} \begin{pmatrix} \left(W^{(\ell+1)}\right)^\top H^{(\ell+1),[t]} A_\ell^{\mathrm{out}} \\ \left(B^{(\ell+1)}\right)^\top H^{(\ell+1),[t]} A_\ell^{\mathrm{out}} \end{pmatrix}.$$

The original coordinates have the same normalization as the narrow backward aggregate only if $N_\ell / N_{\ell+1} = n_\ell / n_{\ell+1}$. The padded coordinates vanish only under the additional condition that $\left(B^{(\ell+1)}\right)^\top H^{(\ell+1),[t]} = 0$ for the relevant hidden states, for instance if $B^{(\ell+1)} = 0$. Thus the bidirectional matrix-product GMN can be made compatible only with a restricted zero-padding subclass, not with the full equivalence of Appendix A.2.

Graph-level readout. Even if the message-passing part preserved zero-padded hidden states (e.g. in the forward matrix-product GMN), the readout stage would still conflict with zero-padding, because zero-padding and duplication pull the pooling normalization in opposite directions. After the pointwise ψ_r, consider a pooling rule $P_n(x_1, \ldots, x_n) = c_n \sum_{i \in [n]} x_i$. Invariance under zero-padding alone forces $c_{n+m} = c_n$ for every m, i.e. c_n is constant in n, which is exactly sum pooling. Invariance under duplication alone forces $c_{kn} = c_n / k$, i.e. $c_n \propto 1/n$, which is exactly mean pooling. A rule invariant under both must satisfy $c_n = c_n / k$ for every k, i.e. $c_n \equiv 0$, so sum pooling and mean pooling are the two invariances taken separately, and no nontrivial pooling rule realizes both at once. We keep the mean because duplication is the structure we target for the reasons given in Appendix A.2, at the cost of readout compatibility with zero-padding.

B.5.1 LIPSCHITZ SENSITIVITY OF GMNS TO THEIR PARAMETERS.

We isolate the parameter-sensitivity estimate used in the uniform-convergence step of Proposition 20, because it is largely a bookkeeping exercise and is merely for technical convenience. Readers can feel free to skip this subsection on a first reading.

We work with the matrix-product GMN of (7), that is, the simple GMN (2) of Section 2.1 carrying the three modifications of Section 3 with the two message maps replaced by (7). Its learned parameter collects the parameters of the four shared MLPs $\psi_v, \psi_h, \psi_r, \rho$ together with the aggregation matrices of (7),

$$\Theta = \left(\vartheta_v, \vartheta_h, \vartheta_r, \vartheta_\rho, (A_\ell^{\text{in}}, A_\ell^{\text{out}})_{\ell=0}^{L+1}\right) \in \mathcal{A} \subset \mathbb{R}^q,$$

where $\mathcal{A}$ is compact.

We first prove that MLPs are Lipschitz in their parameters, uniformly over a compact parameter set.

Lemma 11. *Let $\psi^\vartheta \colon \mathbb{R}^a \to \mathbb{R}^b$ be an MLP with fixed architecture and Lipschitz activations. For every compact parameter set $\mathcal{A}_\psi$, there is $A_{\mathcal{A}_\psi} \geq 1$ such that, for all $\vartheta, \widetilde{\vartheta} \in \mathcal{A}_\psi$ and $u, \widetilde{u} \in \mathbb{R}^a$,*

$$\left\|\psi^\vartheta(u) - \psi^\vartheta(\widetilde{u})\right\|_2 \leq A_{\mathcal{A}_\psi}\|u - \widetilde{u}\|_2, \qquad \left\|\psi^\vartheta(u) - \psi^{\widetilde{\vartheta}}(u)\right\|_2 \leq A_{\mathcal{A}_\psi}\left(1 + \|u\|_2\right)\left\|\vartheta - \widetilde{\vartheta}\right\|_2.$$

Proof. Following the notation of (1), write ψ^ϑ as an L-layer MLP: with $x^{(0)}(u) = u$,

$$h^{(\ell)}(u) = W^{(\ell)}x^{(\ell-1)}(u) + b^{(\ell)}, \qquad x^{(\ell)}(u) = \sigma\left(h^{(\ell)}(u)\right), \qquad \ell = 1, \ldots, L+1,$$

where $\vartheta = \left((W^{(\ell)}, b^{(\ell)})\right)_{\ell=1}^{L+1}$, and $\psi^\vartheta(u) = h^{(L+1)}(u)$. We write $\widetilde{W}^{(\ell)}, \widetilde{b}^{(\ell)}, \widetilde{h}^{(\ell)}(u), \widetilde{x}^{(\ell)}(u)$ for the same quantities with ϑ replaced by $\widetilde{\vartheta}$.

Compactness of $\mathcal{A}_\psi$ gives a constant $B \geq 1$ with $\left\|W^{(\ell)}\right\| \leq B$ and $\left\|b^{(\ell)}\right\|_2 \leq B$ for every layer ℓ and every $\vartheta \in \mathcal{A}_\psi$. Let $\lambda \geq 1$ be a Lipschitz constant of σ and, by the triangle inequality, $\|\sigma(z)\|_2 \leq \lambda\|z\|_2 + \|\sigma(0)\|_2$.

We prove by induction on ℓ that there are constants $C_\ell \geq 1$, depending only on $\mathcal{A}_\psi$ and the architecture, such that for every $\ell = 0, \ldots, L$, all $\vartheta, \widetilde{\vartheta} \in \mathcal{A}_\psi$, and all $u, \widetilde{u} \in \mathbb{R}^a$,

$$\begin{aligned}
\left\|x^{(\ell)}(u)\right\|_2 &\leq C_\ell\left(1 + \|u\|_2\right), \\
\left\|x^{(\ell)}(u) - x^{(\ell)}(\widetilde{u})\right\|_2 &\leq C_\ell\|u - \widetilde{u}\|_2, \\
\left\|x^{(\ell)}(u) - \widetilde{x}^{(\ell)}(u)\right\|_2 &\leq C_\ell\left(1 + \|u\|_2\right)\left\|\vartheta - \widetilde{\vartheta}\right\|_2.
\end{aligned}$$

Base case $\ell = 0$. We have $x^{(0)}(u) = \widetilde{x}^{(0)}(u) = u$, so the three left-hand sides are $\|u\|_2$, $\|u - \widetilde{u}\|_2$ and 0, and $C_0 = 1$ suffices.

Inductive step. Assume the three bounds at $\ell - 1$.

The first follows from $\left\|x^{(\ell)}(u)\right\|_2 \leq \lambda\left\|h^{(\ell)}(u)\right\|_2 + \|\sigma(0)\|_2$ together with $\left\|h^{(\ell)}(u)\right\|_2 \leq B\left\|x^{(\ell-1)}(u)\right\|_2 + B$, and then the inductive growth bound on $x^{(\ell-1)}(u)$.

The second follows because $\left\|x^{(\ell)}(u) - x^{(\ell)}(\widetilde{u})\right\|_2 \leq \lambda B\left\|x^{(\ell-1)}(u) - x^{(\ell-1)}(\widetilde{u})\right\|_2$, to which the inductive Lipschitz bound applies.

For the third, we first write

$$h^{(\ell)}(u) - \widetilde{h}^{(\ell)}(u) = W^{(\ell)}\left(x^{(\ell-1)}(u) - \widetilde{x}^{(\ell-1)}(u)\right) + \left(W^{(\ell)} - \widetilde{W}^{(\ell)}\right)\widetilde{x}^{(\ell-1)}(u) + \left(b^{(\ell)} - \widetilde{b}^{(\ell)}\right).$$

The first term satisfies

$$\left\|W^{(\ell)}\left(x^{(\ell-1)}(u) - \widetilde{x}^{(\ell-1)}(u)\right)\right\|_2 \leq B\left\|x^{(\ell-1)}(u) - \widetilde{x}^{(\ell-1)}(u)\right\|_2.$$

The second term satisfies

$$\left\|\left(W^{(\ell)} - \widetilde{W}^{(\ell)}\right)\widetilde{x}^{(\ell-1)}(u)\right\|_2 \leq \left\|\vartheta - \widetilde{\vartheta}\right\|_2\left\|\widetilde{x}^{(\ell-1)}(u)\right\|_2.$$

Finally, the third term satisfies

$$\left\|b^{(\ell)} - \widetilde{b}^{(\ell)}\right\|_2 \le \left\|\vartheta - \widetilde{\vartheta}\right\|_2.$$

Combining these three estimates and applying the inductive bounds gives the desired parameter-difference estimate for $h^{(\ell)}$. Applying σ costs another factor λ.

All three bounds therefore hold at ℓ with $C_\ell = \lambda\left((B+1)C_{\ell-1} + B + 1\right) + \|\sigma(0)\|_2$, which completes the induction.

Finally, $\psi^\vartheta(u) = h^{(L+1)}(u) = W^{(L+1)}x^{(L)}(u) + b^{(L+1)}$, so the three estimates for the affine step above, applied at $\ell = L$ and without the activation, give the two claims of the lemma with $A_{\mathcal{A}_\psi} = (B+1)C_L + 1$. □

We can now prove the parameter-sensitivity estimate for the matrix-product GMN, whose MLP components $\psi_v, \psi_h, \psi_r, \rho$ are controlled by Lemma 11.

Proposition 12. *Assume that the MLP components $\psi_v, \psi_h, \psi_r, \rho$ of the matrix-product GMN have fixed architectures and Lipschitz activations. Then, for every width n and every $\theta_n \in V_n$,*

$$\left\|f_n^\Theta(\theta_n) - f_n^{\widetilde{\Theta}}(\theta_n)\right\|_2 \le c_{\mathcal{A},T}\left(1 + \|\theta_n\|_{V_n,\mathrm{op}}\right)^{T+1}\left\|\Theta - \widetilde{\Theta}\right\|_2, \tag{20}$$

for all $\Theta, \widetilde{\Theta} \in \mathcal{A}$, where $c_{\mathcal{A},T}$ is independent of the hidden widths.

Proof. We first record the uniform MLP estimates that will be used for every shared map. By Lemma 11, applied to the projections of the compact set $\mathcal{A}$ onto the parameter spaces of $\psi_v, \psi_h, \psi_r, \rho$, there is $A_{\mathcal{A}} \ge 1$ such that, for every $\psi \in \{\psi_v, \psi_h, \psi_r, \rho\}$,

$$\left\|\psi^\Theta(u) - \psi^\Theta(\widetilde{u})\right\|_2 \le A_{\mathcal{A}}\|u - \widetilde{u}\|_2, \qquad \left\|\psi^\Theta(u) - \psi^{\widetilde{\Theta}}(u)\right\|_2 \le A_{\mathcal{A}}\left(1 + \|u\|_2\right)\left\|\Theta - \widetilde{\Theta}\right\|_2, \tag{21}$$

for all $\Theta, \widetilde{\Theta} \in \mathcal{A}$ and all inputs $u, \widetilde{u}$ of the corresponding MLP. Here the norm of the MLP-parameter difference is bounded by $\left\|\Theta - \widetilde{\Theta}\right\|_2$ because each MLP parameter is a coordinate subvector of Θ.

Fix a width n and $\theta_n \in V_n$, and put

$$R_n := \|\theta_n\|_{V_n,\mathrm{op}}.$$

Fix $\Theta, \widetilde{\Theta} \in \mathcal{A}$, decorate with a tilde every quantity computed from $\widetilde{\Theta}$, and set

$$M_t^{(\ell)} := \frac{\left\|H^{(\ell),[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}}, \quad \Delta_t^{(\ell)} := \frac{\left\|H^{(\ell),[t]} - \widetilde{H}^{(\ell),[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}}, \quad M_t := \sum_{\ell=0}^{L+1} M_t^{(\ell)}, \quad D_t := \sum_{\ell=0}^{L+1} \Delta_t^{(\ell)},$$

with the conventions $M_t^{(-1)} = M_t^{(L+2)} = 0$. Let

$$a_{\mathcal{A}} := 1 \vee \sup_{\Theta\in\mathcal{A}} \max_{0\le\ell\le L+1} \max\left\{\left\|A_\ell^{\mathrm{in}}\right\|, \left\|A_\ell^{\mathrm{out}}\right\|\right\}, \qquad \kappa_{\mathcal{A}} := \max_{\psi\in\{\psi_v,\psi_h,\psi_r,\rho\}} \sup_{\Theta\in\mathcal{A}} \left\|\psi^\Theta(0)\right\|_2,$$

both finite because $\mathcal{A}$ is compact and, by (21), $\Theta \mapsto \psi^\Theta(0)$ is continuous. Finally, since the normalized spectral norm (5) dominates each of its summands,

$$\sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W^{(\ell)}\right\| \le R_n, \qquad \frac{\left\|b^{(\ell)}\right\|_2}{\sqrt{n_\ell}} \le R_n \qquad (1 \le \ell \le L+1). \tag{22}$$

We prove by induction on the round t that

$$M_t, \widetilde{M}_t \le K_t, \qquad D_t \le C_t\left\|\Theta - \widetilde{\Theta}\right\|_2, \tag{23}$$

for finite constants K_t, C_t given by the explicit recursions (26), (27) below, which depend only on $\mathcal{A}, t, L, d_{\mathrm{in}}, d_{\mathrm{out}}, d_h$ and on R_n.

A row-wise estimate. Let $\psi \in \{\psi_v, \psi_h, \psi_r, \rho\}$ take J arguments, and apply it to the rows of matrices $X_1, \dots, X_J$ with m rows each. Write $u_i := (X_{1,i:}, \dots, X_{J,i:})$ for the ith stacked argument. Since

$\|u_i\|_2 \le \sum_j \|X_{j,i:}\|_2$, the two bounds of (21) hold row by row. Applying the triangle inequality in $\mathbb{R}^m$ to the vectors of row-wise bounds and dividing by $\sqrt{m}$ gives

$$\begin{aligned} \frac{\left\|\psi^{\Theta}(X_1,\dots,X_J)\right\|_{\mathrm{F}}}{\sqrt{m}} &\le \kappa_{\mathcal{A}} + A_{\mathcal{A}} \sum_{j=1}^{J} \frac{\|X_j\|_{\mathrm{F}}}{\sqrt{m}}, \\ \frac{\left\|\psi^{\Theta}(X_1,\dots,X_J) - \psi^{\widetilde{\Theta}}(Y_1,\dots,Y_J)\right\|_{\mathrm{F}}}{\sqrt{m}} & \\ \le A_{\mathcal{A}} \sum_{j=1}^{J} \frac{\|X_j - Y_j\|_{\mathrm{F}}}{\sqrt{m}} &+ A_{\mathcal{A}} \left(1 + \sum_{j=1}^{J} \frac{\|X_j\|_{\mathrm{F}}}{\sqrt{m}}\right) \left\|\Theta - \widetilde{\Theta}\right\|_2. \end{aligned} \tag{24}$$

Base case ($t = 0$). The input nodes carry a constant that does not depend on Θ, so $\Delta_0^{(0)} = 0$ and $M_0^{(0)} = \widetilde{M}_0^{(0)}$ is a fixed number depending only on d_{in} and d_h. For $\ell \ge 1$ the initial features are ψ_v^{Θ} applied to the rows of $b^{(\ell)}$, so (24) and (22) give

$$M_0^{(\ell)} \le \kappa_{\mathcal{A}} + A_{\mathcal{A}} R_n, \qquad \Delta_0^{(\ell)} \le A_{\mathcal{A}} \left(1 + R_n\right) \left\|\Theta - \widetilde{\Theta}\right\|_2.$$

Summing over ℓ gives (23) at $t = 0$ with

$$K_0 := M_0^{(0)} + (L+1)\left(\kappa_{\mathcal{A}} + A_{\mathcal{A}} R_n\right), \qquad C_0 := (L+1) A_{\mathcal{A}} \left(1 + R_n\right).$$

Induction step: feature norms. The aggregate bound (18), with R there replaced by R_n, reads

$$\frac{\left\|Z^{(\ell),\mathrm{in},[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}} \le a_{\mathcal{A}} R_n M_t^{(\ell-1)}, \qquad \frac{\left\|Z^{(\ell),\mathrm{out},[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}} \le a_{\mathcal{A}} R_n M_t^{(\ell+1)}. \tag{25}$$

Feeding this into the first line of (24) for the node update ψ_h,

$$M_{t+1}^{(\ell)} \le \kappa_{\mathcal{A}} + A_{\mathcal{A}} \left(M_t^{(\ell)} + a_{\mathcal{A}} R_n M_t^{(\ell-1)} + a_{\mathcal{A}} R_n M_t^{(\ell+1)}\right),$$

and summing over ℓ gives the scalar recurrence $M_{t+1} \le (L+2)\kappa_{\mathcal{A}} + A_{\mathcal{A}}(1 + 2a_{\mathcal{A}} R_n) M_t$, and likewise for $\widetilde{M}_{t+1}$. So (23) holds at round $t+1$ with

$$K_{t+1} := (L+2)\kappa_{\mathcal{A}} + A_{\mathcal{A}} \left(1 + 2a_{\mathcal{A}} R_n\right) K_t. \tag{26}$$

Induction step: feature differences. Splitting the difference of two forward aggregates into a parameter part and a feature part,

$$\begin{aligned} Z^{(\ell),\mathrm{in},[t]} - \widetilde{Z}^{(\ell),\mathrm{in},[t]} = W^{(\ell)} &\left(H^{(\ell-1),[t]} - \widetilde{H}^{(\ell-1),[t]}\right) A_\ell^{\mathrm{in}}(\Theta) \\ &+ W^{(\ell)} \widetilde{H}^{(\ell-1),[t]} \left(A_\ell^{\mathrm{in}}(\Theta) - A_\ell^{\mathrm{in}}(\widetilde{\Theta})\right), \end{aligned}$$

and estimating each term as in (25), using for the second that the entries of A_ℓ^{in} are coordinates of Θ, so that $\left\|A_\ell^{\mathrm{in}}(\Theta) - A_\ell^{\mathrm{in}}(\widetilde{\Theta})\right\| \le \left\|\Theta - \widetilde{\Theta}\right\|_2$,

$$\frac{\left\|Z^{(\ell),\mathrm{in},[t]} - \widetilde{Z}^{(\ell),\mathrm{in},[t]}\right\|_{\mathrm{F}}}{\sqrt{n_\ell}} \le a_{\mathcal{A}} R_n \Delta_t^{(\ell-1)} + R_n \widetilde{M}_t^{(\ell-1)} \left\|\Theta - \widetilde{\Theta}\right\|_2.$$

The same computation on $(W^{(\ell+1)})^\top$ bounds the backward discrepancy by the same expression with $\Delta_t^{(\ell+1)}$ and $\widetilde{M}_t^{(\ell+1)}$ in place of $\Delta_t^{(\ell-1)}$ and $\widetilde{M}_t^{(\ell-1)}$. Feeding these and (25) into the second line of (24) for ψ_h,

$$\begin{aligned} \Delta_{t+1}^{(\ell)} \le A_{\mathcal{A}} &\left(\Delta_t^{(\ell)} + a_{\mathcal{A}} R_n \left(\Delta_t^{(\ell-1)} + \Delta_t^{(\ell+1)}\right)\right) \\ &+ A_{\mathcal{A}} \left(1 + M_t^{(\ell)}\right) \left\|\Theta - \widetilde{\Theta}\right\|_2 \\ &+ A_{\mathcal{A}}(1 + a_{\mathcal{A}}) R_n \left(M_t^{(\ell-1)} + M_t^{(\ell+1)} + \widetilde{M}_t^{(\ell-1)} + \widetilde{M}_t^{(\ell+1)}\right) \left\|\Theta - \widetilde{\Theta}\right\|_2. \end{aligned}$$

Summing over ℓ, each of $\sum_\ell M_t^{(\ell\pm1)}$ and $\sum_\ell \widetilde{M}_t^{(\ell\pm1)}$ is at most K_t by the induction hypothesis, so

$$D_{t+1} \le A_{\mathcal{A}}\,(1+2a_{\mathcal{A}}R_n)\,D_t + A_{\mathcal{A}}\,((L+2)+(1+4(1+a_{\mathcal{A}})R_n)\,K_t)\left\|\Theta-\widetilde{\Theta}\right\|_2,$$

which is (23) at round $t+1$ with

$$C_{t+1} := A_{\mathcal{A}}\,(1+2a_{\mathcal{A}}R_n)\,C_t + A_{\mathcal{A}}\,((L+2)+(1+4(1+a_{\mathcal{A}})R_n)\,K_t)\,. \tag{27}$$

This completes the induction, which we run up to the last round $t=T$.

Degree of the constants. Both (26) and (27) are affine recursions whose coefficients are bounded by a multiple of $1+R_n$, and $K_0, C_0 \le c(1+R_n)$. Hence, by induction on t, $K_t, C_t \le c_t(1+R_n)^{t+1}$ with c_t depending only on $\mathcal{A}, t, L, d_{\text{in}}, d_{\text{out}}, d_h$; in particular

$$K_T, C_T \le c_T\,(1+R_n)^{T+1}\,. \tag{28}$$

Readout. By the layer-wise mean readout of Section 3, the argument of ρ^Θ is $u_\Theta := \left(r_1,\dots,r_L,(h_i^{(0),[T]})_{i\in[d_{\text{in}}]},(h_i^{(L+1),[T]})_{i\in[d_{\text{out}}]}\right)$ with $r_\ell = \frac{1}{n_\ell}\sum_{i\in[n_\ell]}\psi_r^\Theta\left(h_i^{(\ell),[T]}\right)$. Applying (21) to ψ_r row by row and then Cauchy–Schwarz, $\sum_{i\in[n_\ell]} c_i \le \sqrt{n_\ell}\left(\sum_i c_i^2\right)^{1/2}$, gives for $1\le\ell\le L$

$$\|r_\ell\|_2 \le \kappa_{\mathcal{A}} + A_{\mathcal{A}}M_T^{(\ell)}, \qquad \|r_\ell - \widetilde{r}_\ell\|_2 \le A_{\mathcal{A}}\Delta_T^{(\ell)} + A_{\mathcal{A}}\left(1+M_T^{(\ell)}\right)\left\|\Theta-\widetilde{\Theta}\right\|_2.$$

The remaining arguments are listed coordinatewise, and $\left\|(h_i^{(0),[T]})_i\right\|_2 = \sqrt{d_{\text{in}}}M_T^{(0)}$ and $\left\|(h_i^{(0),[T]})_i - (\widetilde{h}_i^{(0),[T]})_i\right\|_2 = \sqrt{d_{\text{in}}}\Delta_T^{(0)}$, and similarly at $\ell = L+1$ with d_{out}. Summing the $L+2$ blocks and using (23),

$$\|u_\Theta\|_2 \le c\,(1+K_T)\,, \qquad \left\|u_\Theta - u_{\widetilde{\Theta}}\right\|_2 \le c\,(C_T+1+K_T)\left\|\Theta-\widetilde{\Theta}\right\|_2,$$

with c depending only on $\mathcal{A}, L, d_{\text{in}}, d_{\text{out}}$. Applying (21) to ρ once more,

$$\left\|f_n^\Theta(\theta_n) - f_n^{\widetilde{\Theta}}(\theta_n)\right\|_2 \le A_{\mathcal{A}}\left\|u_\Theta - u_{\widetilde{\Theta}}\right\|_2 + A_{\mathcal{A}}\,(1+\|u_\Theta\|_2)\left\|\Theta-\widetilde{\Theta}\right\|_2 \le B_n\left\|\Theta-\widetilde{\Theta}\right\|_2,$$

where $B_n := c'\,(1+K_T+C_T)$ is finite and, by (28), at most $c''(1+R_n)^{T+1}$.

Conclusion. Taking $c_{\mathcal{A},T} = c''$ gives (20). □

C Implementations of transferable GMNs

The two recipes of Section 3 were stated for the minimal GMN (2). This section applies them to three published GMN variants, whose design choices are more elaborate: the GMN of Lim et al. (2024) (Appendix C.1), the NG-GNN of Kofinas et al. (2024) (Appendix C.2), and the ScaleGMN of Kalogeropoulos et al. (2024) (Appendix C.3). We do not repeat the proofs of Propositions 1 and 3 for each variant, because they follow the same ideas, as we explain below.

General recipe for duplication-compatible GMNs. The proof of Proposition 1 is an induction over the message-passing rounds, showing at each step that the node features, and respectively the edge features, on the widened network's graph are copies of the corresponding features on the narrow network's graph.

Initial node and edge features. The modification needed is to initialize the edge features with the fan-in rescaled weights $\overline{W}^{(\ell)} = n_{\ell-1}W^{(\ell)}$ instead of the raw weights. Further preprocessing of these scalars needs no modification as long as it is applied entrywise and shared across parameters of the same type. Additional features not read off a single weight or bias entry have to be checked one at a time: we do so in the later sections for the positional encodings and for the probe features of the NG-GNN.

Message and update maps. These are shared across all nodes, or all edges, of a given type, and therefore generally need no modification: equal arguments fed to a shared map return equal outputs

at every copy. All that is required is that the only graph-dependent quantities they read are node, edge, and global features that already satisfy the inductive hypothesis.

Aggregation. This is the step that needs modification, being the only GMN component whose inputs may grow with the width. An aggregation over a set of fixed cardinality is harmless; this covers the input and output layers, whose widths $n_0 = d_{\text{in}}$ and $n_{L+1} = d_{\text{out}}$ are fixed, and any combination over the $L + 1$ layers. An aggregation whose number of terms grows with the width has to be made (i) layer-wise and direction-wise, and (ii) invariant under repeating every element of the aggregated multiset k times. Requirement (ii) rules out the sum and admits the mean, the minimum, the maximum, and the population standard deviation. Requirement (i) is needed because distinct layers may be widened by distinct multipliers k_ℓ.

Graph-level readout. Similarly, the graph-level readout needs to be modified following the same requirement as the local aggregation stated above.

General recipe for matrix-product GMNs. On top of a duplication-compatible GMN, the matrix-product recipe requires an audit of every path through which the input weights or a width-growing collection can influence the computation.

Usually, the weight matrices $(W^{(\ell)})$ enter the computation of a GMN through initial edge features. In this case, three modifications are needed. First, drop all learned hidden-edge features together with the hidden-edge update, so that a hidden weight edge carries only the fan-in rescaled scalar $\overline{W}_{ij}^{(\ell)}$, unchanged across rounds. Second, for messages along a weight edge, restrict the message to multiplication of the weight scalar by a shared linear map of the source state alone, dropping any dependence on the destination state, on global features, and on learned edge states. Third, use layer-wise means for local aggregation and for any global update or readout involving a width-growing layer. Together these restrictions put every appearance of the weight matrices in the form $W^{(\ell)}H^{(\ell-1)}$, or, up to the backward normalization, $\left(W^{(\ell)}\right)^\top H^{(\ell)}$ in the backward direction, where $H^{(\ell)}$ has rows equal to the layer-ℓ node features.

More generally, any other GMN component that directly reads weight-matrix entries must also be modified so that the weight matrices enter only through these matrix products. This single property is what both parts of Proposition 3 use. For compatibility, the induction of Proposition 1 now has to respect a block-wise widening (6), whose blocks $W^{\uparrow}{}_{ij}^{(\ell)}$ are no longer constant. If the backward aggregate is used, it respects only the block-wise widenings satisfying the column-balance condition of Proposition 3(ii). For continuity, the induction propagates bounds on the width-normalized node states $\left\|H^{(\ell),[t]}\right\|_{\text{F}}/\sqrt{n_\ell}$ and on their discrepancies between two weights, and a matrix product is exactly what the normalized spectral norm controls in these terms:

$$\frac{\left\|W^{(\ell)}X\right\|_{\text{F}}}{\sqrt{n_\ell}} \le \sqrt{\frac{n_{\ell-1}}{n_\ell}}\left\|W^{(\ell)}\right\| \cdot \frac{\|X\|_{\text{F}}}{\sqrt{n_{\ell-1}}},$$

the first factor on the right being the layer-ℓ weight term of $\|\cdot\|_{V_n,\text{op}}$ in (5), while shared row-wise Lipschitz maps and layer-wise means are Lipschitz in the same quantities. For the continuity property, all shared maps must be Lipschitz with constants independent of the hidden widths, either globally or uniformly on the on a fixed-radius ball. This holds for most common neural operations; normalization layers are the one case worth spelling out, and we do so where they occur, in Appendix C.2.

C.1 Modifications on the GMN of Lim et al. (2024)

The backbone. The precise mathematical construction of the GMN of Lim et al. (2024) on MLP is given in Dayan et al. (2026) (Definition D.10). We follow their notation to describe the precise modifications needed. The main departure from the simple GMN (2) is the following:

1. Biases are edges rather than node features: there is one bias node $\beta^{(\ell)}$ per layer, joined to every neuron of layer ℓ by an edge carrying $b_i^{(\ell)}$, so the edge set splits into weight edges and bias edges, but both are handled by the same message map, distinguished only by the parameter-type feature described below.

2. Nodes and edges carry additional initial features besides the weight or bias scalar. Additional node features include layer index and node type one-hot vectors, and additional edge features include layer index, direction, and parameter type one-hot vectors.
3. The model maintains a global feature u, and is updated at each round.
4. The graph is undirected. Each edge has two different edge features, one for each direction, with the direction encoded as edge features. Message passing occurs in both directions sharing the same message map.

Duplication-compatible GMN. We now spell out the model obtained by applying the three modifications of Proposition 1 to this backbone. We only state here the modifications needed on top of the formulation of Definition D.10 of Dayan et al. (2026).

Graph. We need to modify the graph construction from an undirected graph to a directed one, since during message-passing, the messages in the forward and backward directions needed to be aggregated separately.

Concretely, the node set is $N = \bigcup_{\ell=0}^{L+1}\{v_i^{(\ell)} : i \in [n_\ell]\} \cup \bigcup_{\ell=1}^{L+1}\{\beta^{(\ell)}\}$, one neuron node per neuron and one bias node per layer, and the edge set is $E = E_{\mathrm{w}} \cup E_{\mathrm{b}}$ with

$$E_{\mathrm{w}} = \left\{(v_j^{(\ell-1)}, v_i^{(\ell)}) : \ell \in [L+1], i \in [n_\ell], j \in [n_{\ell-1}]\right\},$$
$$E_{\mathrm{b}} = \left\{(\beta^{(\ell)}, v_i^{(\ell)}) : \ell \in [L+1], i \in [n_\ell]\right\}.$$

Node and edge features. We modify the weight edge features to carry the fan-in rescaled weight $\overline{W}_{ij}^{(\ell)} = n_{\ell-1} W_{ij}^{(\ell)}$; also, we remove the direction edge features, because now the directions are encoded in the graph structure. The node features and global feature do not require any modification. The layer-index and node-type one-hot features are admissible because they are compatible with duplication widening (4): for example, all neurons of a hidden layer receive the same pair of one-hot vectors and hence unchanged when the input networks are widened.

Concretely, the edge features are modified to be

$$e_{(v_j^{(\ell-1)}, v_i^{(\ell)})} = \left(\overline{W}_{ij}^{(\ell)},\, \mathrm{layer}(\ell),\, \mathrm{ptype(w)}\right), \qquad e_{(\beta^{(\ell)}, v_i^{(\ell)})} = \left(b_i^{(\ell)},\, \mathrm{layer}(\ell),\, \mathrm{ptype(b)}\right).$$

Message-passing layer update. In the update of node features, we modify the message passing to separately aggregate the forward and backward messages, and replace sum aggregation with mean aggregation. The update of edge features is unchanged. In the update of the global feature, we replace the sum over all nodes and edges with a mean over each layer separately.

Concretely, a layer is a tuple of MLPs $(\phi_m^{\rightarrow}, \phi_m^{\leftarrow}, \phi_h, \phi_e, \phi_u)$, shared across the graph, updating $(h, e, u) \mapsto (h', e', u')$ by

$$\begin{aligned}
m_{v_j^{(\ell-1)} \to v_i^{(\ell)}} &= \phi_m^{\rightarrow}\left(h_{v_j^{(\ell-1)}}, h_{v_i^{(\ell)}}, e_{(v_j^{(\ell-1)}, v_i^{(\ell)})}, u\right), \\
m_{v_i^{(\ell)} \to v_j^{(\ell-1)}} &= \phi_m^{\leftarrow}\left(h_{v_i^{(\ell)}}, h_{v_j^{(\ell-1)}}, e_{(v_j^{(\ell-1)}, v_i^{(\ell)})}, u\right), \\
m_{\beta^{(\ell)} \to v_i^{(\ell)}} &= \phi_m^{\rightarrow}\left(h_{\beta^{(\ell)}}, h_{v_i^{(\ell)}}, e_{(\beta^{(\ell)}, v_i^{(\ell)})}, u\right), \\
m_{v_i^{(\ell)} \to \beta^{(\ell)}} &= \phi_m^{\leftarrow}\left(h_{v_i^{(\ell)}}, h_{\beta^{(\ell)}}, e_{(\beta^{(\ell)}, v_i^{(\ell)})}, u\right), \\
h'_{v_i^{(\ell)}} &= \phi_h\Big(h_{v_i^{(\ell)}},\ \tfrac{1}{n_{\ell-1}} \sum_{j \in [n_{\ell-1}]} m_{v_j^{(\ell-1)} \to v_i^{(\ell)}},\ \\
&\qquad \tfrac{1}{n_{\ell+1}} \sum_{j \in [n_{\ell+1}]} m_{v_j^{(\ell+1)} \to v_i^{(\ell)}},\ m_{\beta^{(\ell)} \to v_i^{(\ell)}},\ u\Big), \\
h'_{\beta^{(\ell)}} &= \phi_h\left(h_{\beta^{(\ell)}},\ \tfrac{1}{n_\ell} \sum_{i \in [n_\ell]} m_{v_i^{(\ell)} \to \beta^{(\ell)}},\ u\right),
\end{aligned}$$

$$
\begin{aligned}
e'_{ji} &= \phi_e\left(h_j, h_i, e_{ji}, u\right), \\
u' &= \phi_u\Big(\sum_{\ell=0}^{L+1} \tfrac{1}{n_\ell} \sum_{i\in[n_\ell]} h_{v_i^{(\ell)}},\ \sum_{\ell=1}^{L+1} h_{\beta^{(\ell)}}, \\
&\qquad \sum_{\ell=1}^{L+1} \tfrac{1}{n_{\ell-1}n_\ell} \sum_{j,i} e_{(v_j^{(\ell-1)}, v_i^{(\ell)})},\ \sum_{\ell=1}^{L+1} \tfrac{1}{n_\ell} \sum_{i\in[n_\ell]} e_{(\beta^{(\ell)}, v_i^{(\ell)})},\ u\Big),
\end{aligned}
\tag{29}
$$

with the two sums in $h'_{v_i^{(\ell)}}$ set to 0 at $\ell = 0$ and $\ell = L+1$ respectively, where the corresponding neighborhood is empty. The two directions of message passing use different message maps, $\phi_m^{\rightarrow}$ along the edges and $\phi_m^{\leftarrow}$ against them, since the direction edge feature that distinguished them is gone. Within each direction, however, we follow Lim et al. (2024) and share one message map between the messages among neuron nodes and the messages between a bias node and its neuron nodes, and likewise share one update map ϕ_h between neuron nodes and bias nodes. In principle these could instead be specialized per message type or per node type for extra expressiveness, at the cost of additional parameters.

Graph-level readout. The graph-level output is read off $u^{[T]}$ after the final round, optionally through a further learnable map; since u is already assembled from layer-wise means summed above, no separate pooling stage is needed.

Matrix-product GMN. On top of the duplication-compatible model above, the second level lets each weight and each bias act on the adjacent node state by a matrix product as in (7), and removes every remaining component that is neither linear in the weights of one layer nor row-wise in the node states.

Node and edge features. We drop all edge features beyond the raw scalars, together with the edge update ϕ_e, so that a weight edge carries only $\overline{W}_{ij}^{(\ell)}$ and a bias edge only $b_i^{(\ell)}$, held fixed across rounds. The node features and the global feature u are left unchanged.

Message-passing layer update. We make the following changes to the MPNN update equations in (29). First, as we have removed the edge features, the edge update ϕ_e is removed too, and the global update ϕ_u does not use edge features anymore. Second, both message functions $\phi_m^{\rightarrow}$ and $\phi_m^{\leftarrow}$ are restricted to be linear in the edge scalar and in the source state alone, dropping their dependence on the destination state and on the global feature.

Concretely, $\phi_m^{\rightarrow}$ and $\phi_m^{\leftarrow}$ of (29) is restricted to the form below

$$
\phi_m^{\rightarrow}(h_{\mathrm{src}}, h_{\mathrm{dst}}, e, u) = e\, A^{\rightarrow} h_{\mathrm{src}}, \qquad \phi_m^{\leftarrow}(h_{\mathrm{src}}, h_{\mathrm{dst}}, e, u) = e\, A^{\leftarrow} h_{\mathrm{src}},
$$

for learnable $A^{\rightarrow}, A^{\leftarrow} \in \mathbb{R}^{d_h \times d_h}$, shared across layers as in (29) above (unlike the per-layer $A_\ell^{\mathrm{in}}, A_\ell^{\mathrm{out}}$ of (7)). Explicitly, the four messages read

$$
\begin{aligned}
m^{[t]}_{v_j^{(\ell-1)}\to v_i^{(\ell)}} &= \overline{W}_{ij}^{(\ell)}\, A^{\rightarrow} h^{[t]}_{v_j^{(\ell-1)}}, & m^{[t]}_{v_i^{(\ell)}\to v_j^{(\ell-1)}} &= \overline{W}_{ij}^{(\ell)}\, A^{\leftarrow} h^{[t]}_{v_i^{(\ell)}}, \\
m^{[t]}_{\beta^{(\ell)}\to v_i^{(\ell)}} &= b_i^{(\ell)}\, A^{\rightarrow} h^{[t]}_{\beta^{(\ell)}}, & m^{[t]}_{v_i^{(\ell)}\to \beta^{(\ell)}} &= b_i^{(\ell)}\, A^{\leftarrow} h^{[t]}_{v_i^{(\ell)}}.
\end{aligned}
\tag{30}
$$

The global feature is updated by

$$
u^{[t+1]} = \phi_u\left(\sum_{\ell=0}^{L+1} \tfrac{1}{n_\ell} \sum_{i\in[n_\ell]} h^{[t]}_{v_i^{(\ell)}},\ \sum_{\ell=1}^{L+1} h^{[t]}_{\beta^{(\ell)}},\ u^{[t]}\right).
$$

Equivalently this can be written in matrix form. Write $H^{(\ell),[t]} \in \mathbb{R}^{n_\ell \times d_h}$ for the matrix whose ith row is $h^{[t]}_{v_i^{(\ell)}}$. The messages of (30) can be written as matrix products:

$$
\begin{aligned}
Z^{(\ell),\mathrm{in},[t]} &= W^{(\ell)} H^{(\ell-1),[t]} A^{\rightarrow}, & Z^{(\ell),\mathrm{out},[t]} &= \frac{n_\ell}{n_{\ell+1}} \left(W^{(\ell+1)}\right)^{\top} H^{(\ell+1),[t]} A^{\leftarrow}, \\
Z^{(\ell),\mathrm{b},[t]} &= b^{(\ell)} \left(A^{\rightarrow} h^{[t]}_{\beta^{(\ell)}}\right)^{\top}, & z^{(\ell),\mathrm{b},[t]} &= \frac{1}{n_\ell} \left(b^{(\ell)}\right)^{\top} H^{(\ell),[t]} A^{\leftarrow}.
\end{aligned}
\tag{31}
$$

The node and global updates are then, with ϕ_h applied row-wise,

$$H^{(\ell),[t+1]} = \phi_h\left(H^{(\ell),[t]},\ Z^{(\ell),\mathrm{in},[t]},\ Z^{(\ell),\mathrm{out},[t]},\ Z^{(\ell),\mathrm{b},[t]},\ u^{[t]}\right),$$
$$h^{[t+1]}_{\beta^{(\ell)}} = \phi_h\left(h^{[t]}_{\beta^{(\ell)}},\ z^{(\ell),\mathrm{b},[t]},\ u^{[t]}\right),$$
$$u^{[t+1]} = \phi_u\left(\sum_{\ell=0}^{L+1} \tfrac{1}{n_\ell}\mathbf{1}_{n_\ell}^\top H^{(\ell),[t]},\ \sum_{\ell=1}^{L+1} h^{[t]}_{\beta^{(\ell)}},\ u^{[t]}\right).$$

C.2 Modifications on the NG-GNN of Kofinas et al. (2024)

The backbone. We now explicitly describe the NG-GNN of Kofinas et al. (2024) on MLP inputs.

Graph. The graph construction is exactly the same as the simple GMN in the main text. That is, the node set is $N = \bigcup_{\ell=0}^{L+1}\{v_i^{(\ell)} : i \in [n_\ell]\}$ and the edge set is $E = \left\{(v_j^{(\ell-1)}, v_i^{(\ell)}) : \ell \in [L+1], i \in [n_\ell], j \in [n_{\ell-1}]\right\}$.

Node and edge features. The raw node and edge features are standardized versions of the bias and weight scalars, respectively:

$$\widetilde{h}_{v_i^{(\ell)}} = \begin{cases} 0, & \ell = 0, \\ \left(b_i^{(\ell)} - \mu_{b,\ell}\right)/s_{b,\ell}, & 1 \le \ell \le L+1, \end{cases} \qquad \widetilde{e}_{v_j^{(\ell-1)},v_i^{(\ell)}} = \left(W_{ij}^{(\ell)} - \mu_{W,\ell}\right)/s_{W,\ell}, \quad (32)$$

where the mean $\mu_{W,\ell}$ and standard deviation $s_{W,\ell}$ are taken over all training dataset samples and all matrix entries of $W^{(\ell)}$, and likewise for $\mu_{b,\ell}$ and $s_{b,\ell}$.

Then the initial node and edge features are obtained by

$$h^{[0]}_{v_i^{(\ell)}} = A_V\left(P_b(\widetilde{h}_{v_i^{(\ell)}}) + r_{v_i^{(\ell)}}\right) + p_{\tau(v_i^{(\ell)})}, \qquad e^{[0]}_{v_j^{(\ell-1)},v_i^{(\ell)}} = A_E P_W(\widetilde{e}_{v_j^{(\ell-1)},v_i^{(\ell)}}). \quad (33)$$

Here $P_b : \mathbb{R} \to \mathbb{R}^{d_h}$ and $P_W : \mathbb{R} \to \mathbb{R}^{d_h}$ are either linear maps or Gaussian sinusoidal maps $x \mapsto C(\sin(Bx), \cos(Bx))$, with $B \in \mathbb{R}^{m \times q_{\mathrm{in}}}$ Gaussian and $C \in \mathbb{R}^{d_h \times 2m}$ learnable, where $q_{\mathrm{in}} = 1$ here. The maps $A_V, A_E \in \mathbb{R}^{d_h \times d_h}$ are linear maps. The vector $p_{\tau(v)} \in \mathbb{R}^{d_h}$ is a learnable position encoding, with $\tau(v)$ distinguishes input coordinates, hidden layers, and output coordinates: all nodes in a hidden layer $\ell \in \{1, \dots, L\}$ share one type ID, while the input nodes $v_i^{(0)}$ and output nodes $v_i^{(L+1)}$ each have coordinate-specific type IDs.

If q learned probe inputs are used, then the probe feature r_v is obtained by taking the vector in $\mathbb{R}^q$ of activations of node v across those probes and applying a layer-specific linear map $\mathbb{R}^q \to \mathbb{R}^{d_h}$ followed by layer normalization.

Message-passing layer update. The message from node $v_j^{(\ell-1)}$ to node $v_i^{(\ell)}$ is computed in the following two possible ways. In the FiLM variant, the edge features are used to produce scale and shift vectors applied on the messages:

$$\begin{aligned}(\gamma^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}, \beta^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}) &= A_{\mathrm{film}} e^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}, \\ m^{[t]}_{v_j^{(\ell-1)} \to v_i^{(\ell)}} &= \gamma^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}} \odot \eta_t\left(h^{[t]}_{v_i^{(\ell)}}, h^{[t]}_{v_j^{(\ell-1)}}\right) + \beta^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}},\end{aligned} \quad (34)$$

where $A_{\mathrm{film}} \in \mathbb{R}^{2d_h \times d_h}$ is a linear map and $\eta_t : \mathbb{R}^{2d_h} \to \mathbb{R}^{d_h}$ is an MLP. In the non-FiLM variant, we instead use

$$m^{[t]}_{v_j^{(\ell-1)} \to v_i^{(\ell)}} = \eta_t\left(h^{[t]}_{v_i^{(\ell)}}, h^{[t]}_{v_j^{(\ell-1)}}, A_{\mathrm{edge}} e^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}\right),$$

where $A_{\mathrm{edge}} \in \mathbb{R}^{d_h \times d_h}$ is a linear map and $\eta_t : \mathbb{R}^{3d_h} \to \mathbb{R}^{d_h}$ is an MLP.

The node features are updated by

$$h^{[t+1]}_{v_i^{(\ell)}} = \Gamma_t\left(h^{[t]}_{v_i^{(\ell)}}, \mathrm{AGG}_{\mathrm{PNA}}\left(\{m^{[t]}_{v_j^{(\ell-1)} \to v_i^{(\ell)}} : j \in [n_{\ell-1}]\}\right)\right), \qquad \ell \in [L+1], \quad (35)$$

with the aggregate taken to be empty at $\ell = 0$, since $v_i^{(0)}$ has no in-neighbors; $\mathrm{AGG}_{\mathrm{PNA}}$ is the PNA aggregation operator of Corso et al. (2020), which combines multiple different permutation-invariant aggregators, and Γ_t is an MLP with layer normalization and dropout.

The edge features are updated as

$$e^{[t+1]}_{v_j^{(\ell-1)},v_i^{(\ell)}} = A_o^{\mathrm{upd}}\sigma\left(A_e^{\mathrm{upd}} e^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}} + A_s^{\mathrm{upd}} h^{[t+1]}_{v_j^{(\ell-1)}} + A_t^{\mathrm{upd}} h^{[t+1]}_{v_i^{(\ell)}}\right), \tag{36}$$

where $A_o^{\mathrm{upd}}, A_e^{\mathrm{upd}}, A_s^{\mathrm{upd}}, A_t^{\mathrm{upd}} \in \mathbb{R}^{d_h \times d_h}$ are linear maps, and σ is an activation.

Graph-level readout. The final readout is an MLP ρ applied to the concatenation of the output-layer node features:

$$f^{\Theta}(\theta) = \rho\left(\left(h^{[T]}_{v_i^{(L+1)}}\right)_{i\in[n_{L+1}]}\right). \tag{37}$$

We summarize the main departure from the simple GMN (2):

1. The node and edge features use standardization on biases and weights, and optional sinusoidal encodings, positional encodings, and probe features.
2. The message functions take special forms that are either FiLM or non-FiLM.
3. The message passing aggregation is PNA rather than simple sum aggregation.
4. The readout uses only the last-layer node features, rather than all node features.

Duplication-compatible NG-GNN. The modifications needed to make NG-GNN duplication-compatible are the following:

Node and edge features. We modify the raw weight edge feature of (32) to standardize the fan-in rescaled weight $\overline{W}_{ij}^{(\ell)} = n_{\ell-1} W_{ij}^{(\ell)}$ rather than $W_{ij}^{(\ell)}$. The standardization itself is admissible, being entrywise affine with per-layer constants.

The positional encoding $p_{\tau(v)}$ of (33) needs no modification because they are compatible with duplication widening (4): all neurons of a hidden layer share one ID, while the coordinate-specific IDs occur only at the input and output layers, whose widths n_0 and n_{L+1} do not grow.

The probe features need no modification either, since under the duplication widening (4) the widened activations are $x^{(\ell)\uparrow} = x^{(\ell)} \otimes \mathbf{1}_{k_\ell}$, so all copies of a neuron carry the same activation at each of the q probe inputs.

Message-passing layer update. We replace the PNA aggregation of (35) by the restriction $\mathrm{AGG}_{\mathrm{PNA}}^{\mathrm{dup}}$ that keeps only the aggregators invariant under repeating every element of the aggregated multiset k times — the mean, min, max, and standard deviation, but not the sum — and that keeps only the identity degree scaler of Corso et al. (2020), since other scalers are width-dependent.

Dropout can break duplication-compatibility during training, but compatibility holds at evaluation time, when dropout is turned off.

Graph-level readout. The readout (37) is already duplication-compatible, since it reads only the output-layer node features and n_{L+1} does not grow.

Matrix-product NG-GNN. The further modifications needed to make the matrix-product NG-GNN are also almost the same as those for the simple GMN.

Node and edge features. We drop the edge features altogether, together with the edge update (36), so that what is left on an edge is the raw scalar $\overline{W}_{ij}^{(\ell)}$, held fixed across rounds. In particular the map P_W of (33) has to be removed; the standardization of (32) is entrywise affine with per-layer constants and could be kept, but we drop it as well and use the raw scalar.

The probe features can also be kept. Fix the q probe inputs and let $X^{(\ell)} \in \mathbb{R}^{n_\ell \times q}$ hold the layer-ℓ activations of the input network, one row per neuron. The ordinary forward pass computes them by

$$X^{(\ell)} = \sigma\left(\frac{1}{n_{\ell-1}}\overline{W}^{(\ell)} X^{(\ell-1)} + b^{(\ell)}\mathbf{1}_q^\top\right), \qquad 1 \le \ell \le L,$$

and by the same formula without the nonlinearity at $\ell = L + 1$. So each weight matrix acts only by multiplying the previous activation matrix, and each bias only through the rank-one term $b^{(\ell)}\mathbf{1}_q^\top$, which is the matrix-product form we need. The map that turns a row of $X^{(\ell)}$ into the feature r_v is shared across nodes, so it is admissible. In particular, the layer normalization inside this map can be kept unmodified, since the normalization is taken over the hidden dimension d_h which does not grow with the width. When regularized by an ε inside the square root, it is also Lipschitz with a constant depending on its learned scale and shift ε but not on the hidden width.

Message-passing layer update. We restrict $\mathrm{AGG}^{\mathrm{dup}}_{\mathrm{PNA}}$ further to the mean aggregator alone, the only one that is linear in the aggregated messages, and we restrict the message to

$$m^{[t]}_{v_j^{(\ell-1)} \to v_i^{(\ell)}} = \overline{W}^{(\ell)}_{ij}\, A^{\mathrm{in}}_\ell h^{[t]}_{v_j^{(\ell-1)}},$$

which specializes both (34) and its non-FiLM counterpart. Equivalently, this can be written in matrix form. Writing $H^{(\ell),[t]} \in \mathbb{R}^{n_\ell \times d_h}$ for the matrix whose ith row is $h^{[t]}_{v_i^{(\ell)}}$, the mean aggregate and the node update are

$$Z^{(\ell),\mathrm{in},[t]} = W^{(\ell)} H^{(\ell-1),[t]} A^{\mathrm{in}}_\ell, \qquad H^{(\ell),[t+1]} = \Gamma_t\left(H^{(\ell),[t]}, Z^{(\ell),\mathrm{in},[t]}\right),$$

with Γ_t applied row-wise.

The layer normalization applied here needs to be modified. In the original implementation, this normalization is not applied separately to each row in $\mathbb{R}^{d_h}$; its mean and variance are computed from a width-growing collection of node features. After the aggregate $Z^{(\ell),\mathrm{in},[t]} = W^{(\ell)} H^{(\ell-1),[t]} A^{\mathrm{in}}_\ell$, those statistics include global averages and squared terms formed from $Z^{(\ell),\mathrm{in},[t]}$. These terms are nonlinear functions of $W^{(\ell)}$, so the weights no longer enter only by matrix multiplication. We therefore replace this operation with layer normalization over the fixed dimension d_h, or omit it.

C.3 Modifications on the ScaleGMN of Kalogeropoulos et al. (2024)

The backbone. The ScaleGMN of Kalogeropoulos et al. (2024) further imposes scaling symmetries on top of the permutation symmetries. We now explicitly describe the ScaleGMN construction on MLP inputs, and we follow the implementation of ScaleGMN in their experiments, since we will also use it in our experiments.

Graph. The graph construction is exactly the same as the simple GMN in the main text. That is, a directed graph with node set $N = \bigcup_{\ell=0}^{L+1} \{v_i^{(\ell)} : i \in [n_\ell]\}$ and the edge set $E = \left\{(v_j^{(\ell-1)}, v_i^{(\ell)}) : \ell \in [L+1], i \in [n_\ell], j \in [n_{\ell-1}]\right\}$.

Node and edge features. Node and edge features are initialized from the biases and weight entries as in the simple GMN, except that the scalars are first projected to the hidden dimension d_h and then combined with learnable positional encoding in a way that preserves the intended scaling behavior:

$$h^{[0]}_{v_i^{(\ell)}} = \begin{cases} W_V 1 + p_{\tau(v_i^{(0)})}, & \ell = 0, \\ \left(W_V b_i^{(\ell)}\right) \odot p_{\tau(v_i^{(\ell)})}, & 1 \le \ell \le L, \\ W_V b_i^{(L+1)} + p_{\tau(v_i^{(L+1)})}, & \ell = L+1, \end{cases} \qquad e^{[0]}_{v_j^{(\ell-1)}, v_i^{(\ell)}} = \left(W_E W_{ij}^{(\ell)}\right) \odot q_{\tau(v_j^{(\ell-1)}, v_i^{(\ell)})}, \tag{38}$$

where $W_V, W_E \in \mathbb{R}^{d_h}$ are learnable bias-free linear maps $\mathbb{R} \to \mathbb{R}^{d_h}$ (written as a single column vector), and $p_{\tau(v)}, q_{\tau(u,v)} \in \mathbb{R}^{d_h}$ are learnable embeddings indexed by node/edge type IDs $\tau(v)$ and $\tau(u,v)$. Concretely, $\tau(v)$ distinguishes input coordinates, hidden layers, and output coordinates: all nodes in a hidden layer $\ell \in \{1, \dots, L\}$ share one type ID, while the input nodes $v_i^{(0)}$ and output nodes $v_i^{(L+1)}$ each have coordinate-specific type IDs. Likewise, $\tau(u,v)$ depends only on the layer pair and on whether the edge is attached to the input or output boundary: edges $(v_j^{(0)}, v_i^{(1)})$ are typed by the input coordinate j, interior hidden-to-hidden edges $(v_j^{(\ell-1)}, v_i^{(\ell)})$ with $2 \le \ell \le L$ share one type ID per layer pair $(\ell-1, \ell)$, and final-layer edges $(v_j^{(L)}, v_i^{(L+1)})$ are typed by the output coordinate i.

The bidirectional variant of ScaleGMN, where message passing also occurs in the backward direction on the edges, uses a different backward edge feature for positive-scale symmetry. In the formulation of Kalogeropoulos et al. (2024), the backward edge feature for the edge from $v_j^{(\ell-1)}$ to $v_i^{(\ell)}$ is initialized as

$$e^{\leftarrow,[0]}_{v_j^{(\ell-1)},v_i^{(\ell)}} = \left(W_E^{\leftarrow}\left(1/W_{ij}^{(\ell)}\right)\right) \odot q^{\leftarrow}_{\tau(v_j^{(\ell-1)},v_i^{(\ell)})}. \tag{39}$$

For sign symmetry this is unnecessary: the implementation can share the same edge feature as the forward direction, just like in the simple GMN.

Message-passing layer update. For $t = 0, \dots, T-1$, a layer sends a forward message along each edge and a backward message against it, aggregates the two separately, and updates the node state,

$$\begin{aligned}
m^{\rightarrow,[t]}_{v_j^{(\ell-1)}\to v_i^{(\ell)}} &= \Psi_m\left(\left(A_e e^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}\right) \odot \left(A_v h^{[t]}_{v_j^{(\ell-1)}}\right)\right) \in \mathbb{R}^{d_h},\\
m^{\leftarrow,[t]}_{v_j^{(\ell+1)}\to v_i^{(\ell)}} &= \Psi_m^{\leftarrow}\left(\left(A_e^{\leftarrow} e^{\leftarrow,[t]}_{v_i^{(\ell)},v_j^{(\ell+1)}}\right) \odot \left(A_v^{\leftarrow} h^{[t]}_{v_j^{(\ell+1)}}\right)\right) \in \mathbb{R}^{d_h},\\
\widetilde{h}^{[t+1]}_{v_i^{(\ell)}} &= \Psi_h\left(h^{[t]}_{v_i^{(\ell)}}, \sum_{j\in[n_{\ell-1}]} m^{\rightarrow,[t]}_{v_j^{(\ell-1)}\to v_i^{(\ell)}}, \sum_{j\in[n_{\ell+1}]} m^{\leftarrow,[t]}_{v_j^{(\ell+1)}\to v_i^{(\ell)}}\right) \in \mathbb{R}^{d_h},
\end{aligned}$$

where $A_e, A_v, A_e^{\leftarrow}, A_v^{\leftarrow} \in \mathbb{R}^{d_h\times d_h}$ are learnable bias-free linear maps, written as left matrix multiplication on the column vectors $e^{[t]}, h^{[t]} \in \mathbb{R}^{d_h}$, and the two aggregates are set to 0 at $\ell = 0$ and $\ell = L+1$ respectively, where the corresponding neighborhood is empty. The maps $\Psi_m, \Psi_m^{\leftarrow}$ and Ψ_h are the implementation's `EquivariantNet` blocks for the non-permutation symmetry modes: on hidden nodes they apply equivariant transformations for the selected symmetry, while on the fixed input/output nodes they may fall back to ordinary MLP branches. A learnable residual skip connection is then applied

$$h_v^{[t+1]} = S_t h_v^{[t]} + \widetilde{h}_v^{[t+1]},$$

with $S_t \in \mathbb{R}^{d_h\times d_h}$ learnable. This is then followed by entrywise dropout on the node feature matrix.

Both edge states are updated after each GNN layer except the last:

$$\begin{aligned}
e^{[t+1]}_{v_j^{(\ell-1)},v_i^{(\ell)}} &= \underbrace{A_E^{\mathrm{upd}} e^{[t]}_{v_j^{(\ell-1)},v_i^{(\ell)}}}_{\in\mathbb{R}^{d_h}} \odot \underbrace{\phi_E\left(\mathrm{inv}\left(h^{[t]}_{v_j^{(\ell-1)}}\right),\ \mathrm{inv}\left(h^{[t]}_{v_i^{(\ell)}}\right)\right)}_{\in\mathbb{R}^{d_h}},\\
e^{\leftarrow,[t+1]}_{v_i^{(\ell)},v_j^{(\ell+1)}} &= \underbrace{A_E^{\leftarrow,\mathrm{upd}} e^{\leftarrow,[t]}_{v_i^{(\ell)},v_j^{(\ell+1)}}}_{\in\mathbb{R}^{d_h}} \odot \underbrace{\phi_E^{\leftarrow}\left(\mathrm{inv}^{\leftarrow}\left(h^{[t]}_{v_i^{(\ell)}}\right),\ \mathrm{inv}^{\leftarrow}\left(h^{[t]}_{v_j^{(\ell+1)}}\right)\right)}_{\in\mathbb{R}^{d_h}},
\end{aligned} \tag{40}$$

where $A_E^{\mathrm{upd}}, A_E^{\leftarrow,\mathrm{upd}} \in \mathbb{R}^{d_h\times d_h}$ are learnable bias-free linear maps, inv is the invariant canonicalization used by the implementation's `InvariantLayer`, and $\phi_E, \phi_E^{\leftarrow}$ are MLPs, again with independent parameters in the two directions.

Just like simple GMN, this defines the *bidirectional ScaleGMN*, and the *forward ScaleGMN* is the restriction $\Psi_m^{\leftarrow} = 0$, in which the edge states (39), the message parameters $A_e^{\leftarrow}, A_v^{\leftarrow}$, and the backward edge update in (40) are absent.

Graph-level readout. For the implementation's `full_graph` readout, the map has the form

$$f^{\Theta}(\theta) = \rho\left(\sum_{\ell=1}^{L}\sum_{i\in[n_\ell]} g_{\mathrm{hid}}\left(h^{[T]}_{v_i^{(\ell)}}\right),\ \left(h^{[T]}_{v_i^{(0)}}\right)_{i\in[n_0]},\ \left(h^{[T]}_{v_i^{(L+1)}}\right)_{i\in[n_{L+1}]}\right),$$

where g_{hid} is the invariant readout block for the selected symmetry and ρ is the final readout MLP.

A different readout, named `last_layer` in the implementation, reads only the ordered boundary-node features. For the forward and bidirectional variants, it takes the following different forms

$$f^{\Theta,\mathrm{last}}_{\mathrm{fw}}(\theta) = \rho\left(\left(h^{[T]}_{v_i^{(L+1)}}\right)_{i\in[n_{L+1}]}\right),$$

$$f^{\Theta,\text{last}}_{\text{bidir}}(\theta) = \rho\left(W_{\text{last}}\left(h^{[T]}_{v_i^{(L+1)}}\right)_{i\in[n_{L+1}]} + b_{\text{last}},\ \ W_{\text{first}}\left(h^{[T]}_{v_i^{(0)}}\right)_{i\in[n_0]} + b_{\text{first}}\right),$$

where $W_{\text{last}} \in \mathbb{R}^{d_h \times n_{L+1} d_h}$, $W_{\text{first}} \in \mathbb{R}^{d_h \times n_0 d_h}$ (with matching biases) are learnable linear maps that project them down to $\mathbb{R}^{d_h}$ before the final MLP ρ.

We summarize the main departure from the simple GMN (2):

1. Positional encodings are incorporated into the initial node and edge features, in a way that imposes the scaling equivariance.
2. The message and node-update maps take special forms to impose the scaling equivariance.
3. Residual connections and dropouts are used.
4. The bidirectional version on positive-scaling symmetries requires separate backward edge features, which should be initialized with the reciprocal weights.

Duplication-compatible ScaleGMN. The modifications needed to make ScaleGMN duplication-compatible are almost the same as those for the simple GMN.

Node and edge features. We modify the raw weight edge features to carry the fan-in rescaled weight $\overline{W}^{(\ell)}_{ij} = n_{\ell-1} W^{(\ell)}_{ij}$, and, in the bidirectional variant, the backward edge features to carry its reciprocal $1/\overline{W}^{(\ell)}_{ij}$. This does not affect scale equivariance: the fan-in factor $n_{\ell-1}$ is a fixed positive constant, which commutes with the scaling action on the weights.

The type embeddings $p_{\tau(v)}$, $q_{\tau(u,v)}$, and $q^{\leftarrow}_{\tau(u,v)}$ of (38) and (39) need no modification because they are compatible with duplication widening: all neurons of a hidden layer share one ID and all interior hidden-to-hidden edges share one ID per layer pair, while the coordinate-specific IDs occur only at the input and output boundaries, whose widths n_0 and n_{L+1} do not grow.

Message-passing layer update. We replace the sum aggregation in the node-feature update with mean aggregation. Concretely, the update equation is changed to

$$\widetilde{h}^{[t+1]}_{v_i^{(\ell)}} = \Psi_h\left(h^{[t]}_{v_i^{(\ell)}},\ \tfrac{1}{n_{\ell-1}} \sum_{j\in[n_{\ell-1}]} m^{\rightarrow,[t]}_{j\to i},\ \tfrac{1}{n_{\ell+1}} \sum_{j\in[n_{\ell+1}]} m^{\leftarrow,[t]}_{j\to i}\right). \tag{41}$$

Again this does not affect scale equivariance. Dropout can break duplication-compatibility during training, but compatibility holds at evaluation time, when dropout is turned off.

Graph-level readout. The `last_layer` readout is already duplication-compatible. The `full_graph` readout is not, but it can be modified into a layer-wise mean readout, just as in the main text. Specifically, the layer-wise mean readout can be of the form

$$f^{\Theta,\text{layer}}(\theta) = \rho\left(\sum_{\ell=1}^{L} \frac{1}{n_\ell} \sum_{i\in[n_\ell]} g_{\text{hid}}\left(h^{[T]}_{v_i^{(\ell)}}\right),\ \left(h^{[T]}_{v_i^{(0)}}\right)_{i\in[n_0]},\ \left(h^{[T]}_{v_i^{(L+1)}}\right)_{i\in[n_{L+1}]}\right).$$

Scale invariance of the readout is unaffected. Notice that it is fine to either take the sum over layers ℓ, or to instead concatenate the layer-wise mean readouts, as in (3). The latter is more expressive, but it restricts the model to inputs with a fixed number of layers L (inconsequential to us, since we vary only the width).

Matrix-product ScaleGMN. The further modifications needed to make the matrix-product forward ScaleGMN are also almost the same as those for the simple GMN. It is impossible to construct a bidirectional matrix-product ScaleGMN for positive-scaling symmetries, because the backward edge features would need to carry reciprocal weights, which cannot be expressed as a matrix product; for sign symmetries, this obstruction does not arise.

Node and edge features. We drop the edge features altogether, so that what is left is the raw scalar $\overline{W}^{(\ell)}_{ij}$, held fixed across rounds, exactly as in Appendix C.1. This is again harmless for scale equivariance.

Message-passing layer update. We restrict the forward message of (41) by the specialization $A_e = A_v = I_{d_h}$ and $\Psi_m(x) = A_\ell^{\text{in}} x$. Since this is only a specialization of the parameters, the resulting models form a subclass of the duplication-compatible ScaleGMNs above, so scale equivariance is inherited and the only cost is expressivity. Equivalently, this can be written in matrix form. Writing $H^{(\ell),[t]} \in \mathbb{R}^{n_\ell \times d_h}$ for the matrix whose ith row is $h_{v_i^{(\ell)}}^{[t]}$, the mean aggregate and the node update are

$$Z^{(\ell),\text{in},[t]} = W^{(\ell)} H^{(\ell-1),[t]} A_\ell^{\text{in}}, \qquad \widetilde{H}^{(\ell),[t+1]} = \Psi_h\left(H^{(\ell),[t]}, Z^{(\ell),\text{in},[t]}\right). \tag{42}$$

For the bidirectional version under positive-scaling symmetry, this is impossible to construct: the resulting aggregate would be a matrix product in the entrywise reciprocal of $(W^{(\ell+1)})^\top$ rather than in $(W^{(\ell+1)})^\top$ itself.

D EXPERIMENTS

Our experiments are built on the ScaleGMN codebase of Kalogeropoulos et al. (2024). We experiment with four base GMN architectures, all sharing the same graph construction and node/edge preprocessing, obtained by crossing two settings of this codebase, `symmetry` (`permutation` or `scale`) and `direction` (`forward` or `bidirectional`). `symmetry: scale` gives the ScaleGMN exactly as formulated in Appendix C.3 above. `symmetry: permutation` gives the GMN without the scaling symmetries. On top of each of these four base models, we use our recipe to construct a duplication-compatible modification and a matrix-product modification, whenever the latter is well-defined. Table 1 summarizes these model variants and where each is defined.

Table 1: The four base GMN architectures compared in our experiments, and their duplication-compatible and matrix-product variants. The forward and bidirectional ScaleGMN are given explicitly in Appendix C.3, with the modifications specified. The forward and bidirectional GMN are defined analogously.

Base GMN model	**Code setting**	**Duplication-compatible variant**	**Matrix-product variant**
GMN (forward)	`symmetry: permutation` `direction: forward`	1. fan-in rescaled edges 2. mean aggregation 3. layer-wise mean readout	message $\overline{W}_{ij}^{(\ell)} A_\ell^{\text{in}} h_j$.
GMN (bidirectional)	`symmetry: permutation` `direction: bidirectional`		
ScaleGMN (forward)	`symmetry: scale` `direction: forward`		
ScaleGMN (bidirectional)	`symmetry: scale` `direction: bidirectional`		*not defined*

We evaluate all model variants of Table 1 whenever possible, except that the matrix-product bidirectional ScaleGMN is not defined, as explained in Appendix C.3. Moreover, we note that the entire row 4 is not run on the CNN-accuracy prediction task. This is because the bidirectional ScaleGMN has a known numerical instability problem as reported in Appendix A.2 of Kalogeropoulos et al. (2024). A bidirectional positive-scale-equivariant ScaleGMN requires the backward edge feature $1/W$. In these CNN zoos, the weights have very small nonzero entries, and hence their reciprocal values are very large, causing numerical overflows during training.

D.1 VERIFICATION OF GMN COMPATIBILITY AND CONTINUITY

We numerically verify that our implementation of the GMN modifications satisfies the compatibility and continuity properties claimed in Section 3, on the two kinds of input networks used throughout the experiments: MLP weights (MNIST INRs of layout $[2, n, n, 1]$) and CNN weights (the small CIFAR-10 zoo CNN of Unterthiner et al. (2020), three stride-2 convolutions followed by global average pooling and a dense head). All checks use an untrained GMN with $d_h = 32$ and $T = 2$ rounds, in float32. Each table entry is the maximum absolute difference over the sampled input networks, width multipliers, and coordinates; we count a check as passing when this maximum is below 10^{-6}.

Table 2: With all three modifications in place, the modified GMN become duplication-compatible; removing any one of them breaks this property. Entries are the maximum GMN output differences between an input network and its duplication widening (4), the maximum taken over the sampled input networks and over the coordinates of the model's output.

Model	Input	all three modifications	without fan-in rescaling	with sum aggregation	with sum readout (3)
GMN (forward)	MLP	1.8×10^{-7} ✓	4.3×10^{-3} ×	1.3×10^{-2} ×	1.8×10^{-1} ×
GMN (bidirectional)	MLP	2.4×10^{-7} ✓	9.9×10^{-3} ×	3.6×10^{-2} ×	1.2×10^{-1} ×
ScaleGMN (forward)	MLP	1.9×10^{-7} ✓	1.2×10^{-3} ×	6.7×10^{-1} ×	1.5×10^{-1} ×
ScaleGMN (bidirectional)	MLP	3.0×10^{-7} ✓	3.2×10^{-3} ×	5.2×10^{-1} ×	3.1×10^{-2} ×
GMN (forward)	CNN	1.6×10^{-7} ✓	7.4×10^{-4} ×	2.2×10^{-2} ×	1.4×10^{-2} ×
GMN (bidirectional)	CNN	1.6×10^{-7} ✓	9.6×10^{-4} ×	3.3×10^{-2} ×	2.6×10^{-2} ×
ScaleGMN (forward)	CNN	1.2×10^{-7} ✓	3.7×10^{-4} ×	3.4×10^{-2} ×	1.1×10^{-2} ×
ScaleGMN (bidirectional)	CNN	2.6×10^{-7} ✓	4.6×10^{-3} ×	1.8×10^{-2} ×	3.4×10^{-2} ×

Duplication compatibility. We widen each input network by the duplication widening (4) and compare the GMN's output on the widened network against its output on the base network. For the MLP case, we use 5 width-32 INRs, each widened by a pair of hidden-layer multipliers drawn independently and uniformly at random from $k \in \{2, 3, 4\}$. For the CNN case, we use 3 width-16 CNNs, each widened by both $k = 2$ and $k = 3$ (a single multiplier applied to all three convolutional layers), for 6 widened networks in total. Table 2 reports the results, which align with the theoretical prediction of Proposition 1.

Compatibility with block-wise widenings. We next consider the more general, function-preserving widening operations of Appendix A.1, generating wider networks from the following five families:

1. Duplication widening (4).
2. Kronecker widening in Proposition 7 with doubly stochastic $P^{(\ell)}$.
3. Kronecker widening in Proposition 7 with row-stochastic $P^{(\ell)}$.
4. Block-wise widening (6) with balanced blocks (Proposition 3(ii)).
5. General block-wise widening (6).

Table 3 reports the results for the GMN and ScaleGMN families (subtables 3a and 3b), which again align with the theoretical prediction of Proposition 3(ii).

Spectral continuity. We evaluate both models on the family used in the proof of Proposition 2(ii). For MLP weights, we set $\overline{W}^{(2)} = H_n$, the Sylvester Hadamard matrix, and every other weight and bias to zero; since $\|\theta_n\|_{V_n,\mathrm{op}} = 1/\sqrt{n} \to 0$, a model Lipschitz in the normalized spectral norm must satisfy $f(\theta_n) \to f(0)$. For CNN weights, the same family is placed at the center of the second layer's 3×3 kernel, i.e. $\overline{W}^{(2)}[:, :, \mathrm{centre}] = H_n$ with all other weights zero, so that layer acts as a 1×1 convolution with matrix H_n/n. Table 4 reports the results, which align with the theoretical prediction of Proposition 3(i) and Proposition 2(ii).

D.2 Size generalization on hand-crafted widenings

As a sanity check, we first consider size generalization on hand-crafted widenings of the input networks, whose outcome is predicted exactly by the design principles behind the GMN modifications. We train the GMNs at the smallest width only, then evaluate the trained checkpoints on wider input networks obtained by two widening operations, duplication widening (4) and block-wise widening (6), both of which by construction represent exactly the same function as the training-width network.

Dataset and training procedure. Every GMN is trained only on the smallest input width: 51 000 MNIST INRs at width 24, 56 000 FMNIST INRs at width 32, or 8000 CNNs at width $c = 16$ for

Table 3: The forward matrix-product model is compatible with the general block-wise widening; the other four families are subclasses of it and are hence also compatible. The bidirectional matrix-product model is compatible only with block-wise widenings that satisfy the column-balance condition of Proposition 3(ii); the duplication widening and the doubly stochastic Kronecker widening are subclasses of this condition and are hence also compatible. The duplication-compatible model (forward or bidirectional) is compatible only with the duplication widening. The same pattern holds for both the GMN (a) and ScaleGMN families (b). Entries are the output difference between an input network and its widening, maximized over the sampled input networks and over the coordinates of the model's output.

(a) GMN family.

Widenings	Matrix-product (forward)	Matrix-product (bidirectional)	Duplication-compatible (forward)	Duplication-compatible (bidirectional)
MLP weights				
Duplication (4)	2.4×10^{-7} ✓	3.3×10^{-7} ✓	1.8×10^{-7} ✓	2.7×10^{-7} ✓
Kronecker, doubly-stoch.	3.0×10^{-7} ✓	2.4×10^{-7} ✓	4.4×10^{-3} ×	9.8×10^{-3} ×
Kronecker, row-stoch.	2.4×10^{-7} ✓	1.6×10^{-1} ×	2.9×10^{-2} ×	2.5×10^{-2} ×
Block-wise, balanced	2.4×10^{-7} ✓	2.7×10^{-7} ✓	8.9×10^{-3} ×	2.0×10^{-2} ×
Block-wise (6)	2.4×10^{-7} ✓	1.4×10^{-1} ×	3.1×10^{-2} ×	3.0×10^{-2} ×
CNN weights				
Duplication (4)	1.3×10^{-7} ✓	1.9×10^{-7} ✓	1.5×10^{-7} ✓	4.5×10^{-7} ✓
Kronecker, doubly-stoch.	1.2×10^{-7} ✓	2.4×10^{-7} ✓	3.1×10^{-3} ×	4.6×10^{-2} ×
Kronecker, row-stoch.	1.9×10^{-7} ✓	1.5×10^{-2} ×	4.6×10^{-3} ×	3.5×10^{-2} ×
Block-wise, balanced	6.7×10^{-8} ✓	2.4×10^{-7} ✓	3.4×10^{-2} ×	1.6×10^{-1} ×
Block-wise (6)	2.7×10^{-7} ✓	1.4×10^{-1} ×	4.8×10^{-2} ×	2.4×10^{-1} ×

(b) ScaleGMN family.

Widenings	Matrix-product (forward)	Matrix-product (bidirectional)	Duplication-compatible (forward)	Duplication-compatible (bidirectional)
MLP weights				
Duplication (4)	6.6×10^{-7} ✓	7.2×10^{-7} ✓	2.1×10^{-7} ✓	1.8×10^{-7} ✓
Kronecker, doubly-stoch.	4.7×10^{-7} ✓	6.6×10^{-7} ✓	7.6×10^{-2} ×	6.9×10^{-2} ×
Kronecker, row-stoch.	8.6×10^{-7} ✓	6.0×10^{-1} ×	8.8×10^{-2} ×	1.0×10^{-1} ×
Block-wise, balanced	4.8×10^{-7} ✓	7.5×10^{-7} ✓	1.0×10^{-1} ×	7.2×10^{-2} ×
Block-wise (6)	5.0×10^{-7} ✓	4.6×10^{-1} ×	1.0×10^{-1} ×	8.0×10^{-2} ×
CNN weights				
Duplication (4)	2.4×10^{-7} ✓	2.2×10^{-7} ✓	1.8×10^{-7} ✓	3.0×10^{-7} ✓
Kronecker, doubly-stoch.	2.1×10^{-7} ✓	1.2×10^{-7} ✓	6.9×10^{-4} ×	1.6×10^{-3} ×
Kronecker, row-stoch.	1.2×10^{-7} ✓	1.7×10^{-2} ×	5.1×10^{-3} ×	8.7×10^{-3} ×
Block-wise, balanced	2.4×10^{-7} ✓	2.5×10^{-7} ✓	2.1×10^{-2} ×	1.8×10^{-2} ×
Block-wise (6)	4.8×10^{-7} ✓	7.6×10^{-2} ×	1.3×10^{-2} ×	1.6×10^{-2} ×

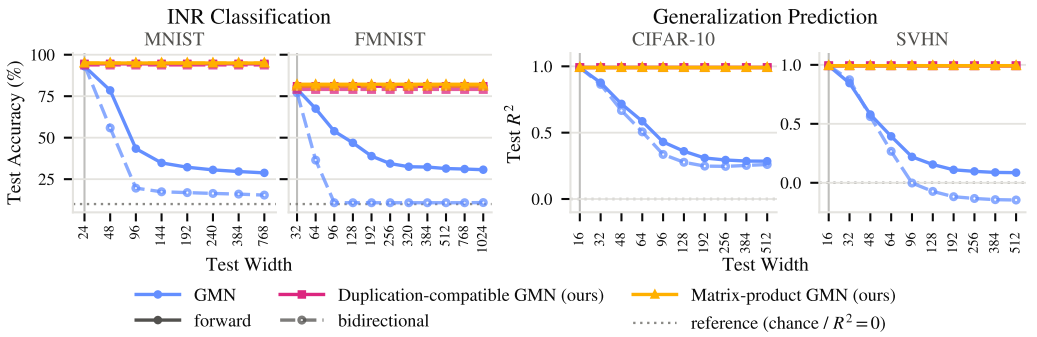
each of the two accuracy-prediction zoos. As test data, we use 1000 held-out networks at the training width. Test networks at larger widths are then obtained by applying the duplication widening (4) or the block-wise widening (6) independently at each target width. Since both widenings preserve the represented function by construction, every widened copy retains its source network's original label.

Training of the GMNs uses AdamW, a batch size of 64, 1000 warmup steps, at most 200 epochs, and early stopping with a patience of 50 epochs. On the INR tasks, the learning rate is chosen by a sweep over the seven values $6.25 \times 10^{-5}, 1.25 \times 10^{-4}, \ldots, 4 \times 10^{-3}$, each candidate trained for 8600 steps ($\approx$ 10 epochs). On the accuracy-prediction tasks, no sweep is run and the configured learning rate of 10^{-3} is used throughout. Both the learning rate and the evaluated checkpoint are selected on validation data at the training width alone.

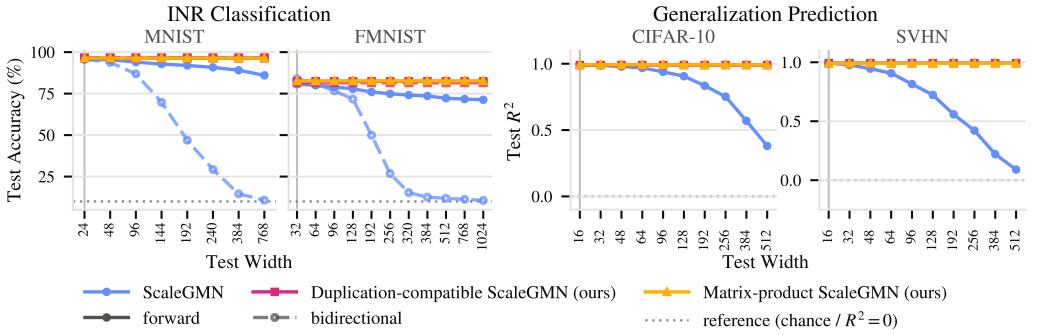
Result 1: Duplication widening. Figures 3a and 3b show the results under the duplication widening (4). As predicted by Proposition 1, the duplication-compatible and matrix-product GMNs repro-

Table 4: The matrix-product GMN is continuous in the normalized spectral norm, whereas the duplication-compatible GMN is not. Entries are $|f(\theta_n) - f(0)|$ on the Hadamard family, which should converge to 0 as $n \to \infty$ for continuous models.

		MLP weights		CNN weights	
n	$\|\theta_n\|_{V_n,\mathrm{op}}$	Matrix-product GMN	Duplication-compatible GMN	Matrix-product GMN	Duplication-compatible GMN
16	0.2500	2.08×10^{-3}	4.99×10^{-3}	8.76×10^{-4}	6.68×10^{-4}
32	0.1768	1.04×10^{-3}	5.05×10^{-3}	4.38×10^{-4}	6.42×10^{-4}
64	0.1250	5.21×10^{-4}	5.08×10^{-3}	2.19×10^{-4}	6.30×10^{-4}
128	0.0884	2.60×10^{-4}	5.09×10^{-3}	1.10×10^{-4}	6.23×10^{-4}
256	0.0625	1.30×10^{-4}	5.10×10^{-3}	—	—



(a) GMN families.

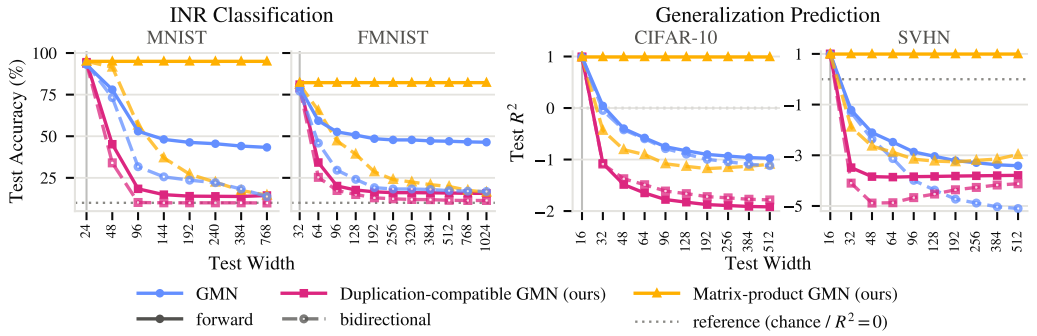


(b) ScaleGMN families. (Bidirectional ScaleGMN is not evaluated on the CNN tasks because of training instability.)

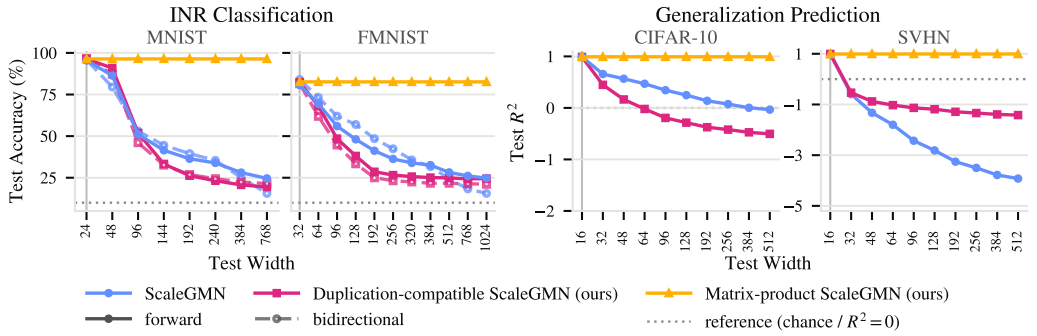
Figure 3: Size generalization on wider input networks generated with duplication widening (4). Both duplication-compatible and matrix-product GMNs generalize in size perfectly, with flat error curves across all test widths, while the baseline GMNs, which are not duplication-compatible, degrade sharply on wider input networks.

duce their in-distribution performance exactly at every test width, confirming that our modifications let the GMNs generalize to wider input networks formed by duplication widening.

Result 2: Block-wise widening. Figures 4a and 4b show the results under the block-wise widening (6). As predicted by Proposition 3(ii), only the forward matrix-product GMNs are compatible with the block-wise widening, and accordingly only they reproduce their in-distribution performance exactly at every test width, confirming that our modifications let them generalize to wider input networks formed by block-wise widening.



(a) GMN families.



(b) ScaleGMN families. (Bidirectional ScaleGMN is not evaluated on the CNN tasks because of training instability.)

Figure 4: Size generalization on wider input networks generated with block-wise widening (6). Only the forward matrix-product GMNs/ScaleGMNs generalize in size perfectly, with flat error curves across all test widths, while all other models, which are not compatible with block-wise widenings, degrade sharply on wider input networks.

D.3 Size generalization on independently trained weights

Next, in the more realistic scenario, we consider size generalization on input networks of different widths that are independently trained from random initialization.

D.3.1 Dataset generation

INR datasets. For each of MNIST and FMNIST, we fit one two-hidden-layer ReLU INR $\mathbb{R}^2 \to \mathbb{R}$ per image, independently at every width, and the metanetwork predicts the image's class from the INR weights alone. The image identities are partitioned across widths before any INR is fitted, so no image contributes weights to more than one width. MNIST uses 51 000 training and 1000 validation INRs at the training width 24, plus 1000 test INRs per width at 13 widths up to 1024 (42.7× the training width). FMNIST uses 56 000 training and 1000 validation INRs at the training width 32, plus 1000 test INRs per width at 12 widths up to 1024 (32×).

The training width is chosen as the smallest width at which the INR fits are of acceptable quality. The reason is that a training width whose fits are markedly worse than those at the test widths would confound the results: a rise in test accuracy with width could then reflect improving input quality rather than genuine size generalization by the metanetwork.

Every width is fitted twice, once under SP and once under μP. μP is set to coincide with SP at the training width, so the two parameterizations agree exactly in distribution and differ only out of distribution. Under SP, the fitting learning rate must be retuned at every width: a learning rate that fits well at the training width tends to perform increasingly poorly at larger widths, yielding

lower-quality inputs there. The selected values decay from 10^{-2} at the training width to 5×10^{-4} (MNIST) and 10^{-3} (FMNIST) at width 1024. Under μP, by contrast, the learning-rate sweep is run only once, at the base width, and the resulting value is held fixed at every other width: Yang et al. (2021) observed that the optimal learning rate is width-independent under μP, which is precisely what allows this hyperparameter transfer across widths.

To assess INR quality, Table 5 reports the peak signal-to-noise ratio (PSNR, measured in decibels, dB; higher is better) and the failure rate ($\mathrm{MSE} > 0.01$) at every fitted width. Under SP, quality peaks a few widths past the training width and then degrades, most severely on FMNIST. Under μP, quality stays essentially flat, with a low failure rate throughout. This is consistent with the finding of Yang et al. (2021) that under μP, wider is simply better, whereas under SP this is usually not the case. This drift in input quality under SP should be taken into account when interpreting the size-generalization results: part of the accuracy decrease at larger widths may be attributable to the decline in input quality rather than to the metanetwork itself. However, the considerable decrease in accuracy under SP at larger widths cannot be explained by input quality alone, so the metanetwork's size generalization can be concluded to be genuinely poor in that regime.

Table 5: Reconstruction quality of the fitted INRs at every width: peak signal-to-noise ratio (PSNR, in dB; higher is better) and failure rate ($\mathrm{MSE} > 0.01$, in %). Under μP, quality generally stays flat across all widths; under SP it degrades past its peak, most severely on FMNIST.

Dataset	Param.	24	32	48	64	80	96	128	192	256	384	512	768	1024
MNIST	SP	30.2/0.7	31.9/0.3	34.5/0.2	34.8/0.2	35.4/0.4	35.9/0.5	35.0/0.4	32.6/0.7	33.4/1.1	31.1/1.1	31.7/1.0	29.3/1.0	29.5/3.2
MNIST	μP	30.2/0.3	31.3/0.1	32.2/0.3	32.8/0.0	33.0/0.0	33.0/0.3	33.2/0.0	33.1/0.1	33.0/0.2	32.8/0.0	32.5/0.0	32.8/0.0	32.5/0.0
FMNIST	SP	—	25.5/5.9	27.3/2.7	27.7/2.8	27.6/2.3	27.9/2.2	28.0/3.9	27.1/4.3	27.0/5.4	25.6/7.5	25.1/10.9	24.1/13.6	23.3/19.5
FMNIST	μP	—	25.5/4.8	26.1/4.4	26.6/3.4	26.9/3.0	27.2/2.3	27.3/1.8	27.3/2.4	27.4/1.9	27.5/1.5	27.4/1.7	27.4/2.2	27.3/2.7

CNN datasets. The two accuracy prediction datasets are zoos of CNNs — three 3×3 convolutions of c channels each, global average pooling, and a linear classifier, 4970 parameters at $c = 16$ — trained on grayscale CIFAR-10 and on grayscale SVHN, and the GMN's task is to predict a member's test accuracy from its weights. Each member draws its hyperparameters log-uniformly (learning rate in $[5 \times 10^{-5}, 5 \times 10^{-2}]$, initialization multiplier in $[0.25, 4]$, weight decay in $[10^{-6}, 10^{-1}]$) together with a dropout rate from $\{0, 0.2, 0.45\}$, a fraction of the training set from $\{0.1, 0.25, 0.5, 1\}$, and an epoch budget from $\{5, 10, 20\}$. Particularly, the hyperparameter distribution is identical at every width and the widths differ only in their number of channels.

We use 8000 training and 1000 validation networks at $c = 16$, and 1000 test networks per width at ten widths from $c = 16$ to $c = 512$ ($32\times$), with a separate seed per role and per width. As on the INR tasks, every width is instantiated twice, under SP and under μP with base width 16. The μP networks are set to coincide with the SP networks at the training width, so the two parameterizations agree exactly in distribution and differ only out of distribution.

We further examine a property of the generated data that matters for the size-generalization evaluation below: how the prediction target itself drifts with width. Wider networks have more capacity and should fit the training data better, so we expect test accuracy to generally increase with width. A GMN trained on the smallest width alone never observes this trend, so extrapolating it is a genuinely out-of-distribution problem. One route by which it could still succeed is if the function computed by a higher-accuracy wide network resembles the function computed by some lower-accuracy narrow network with different hyperparameters that the GMN did see during training, allowing the GMN to generalize by detecting the similarity in the represented function. Table 6 reports, at every width, the fraction of that width's test accuracies lying inside the central 98% range of the training width's accuracies, together with the shift of the median accuracy. It is observed that under SP, there is a clear drift of the prediction target with width, while under μP the target is much more stable. This motivates us to explore various evaluation criteria that are robust to the drift, as described next.

D.3.2 Evaluation and results

Training procedure. The GMNs are trained exactly as in Appendix D.2, except that the learning-rate sweep is always run before selecting the best GMN based on test performance at the training width.

Table 6: Drift of the prediction target with width on the two CNN zoos, relative to the training width $c = 16$. Each cell gives the fraction of that width's test accuracies falling inside the central 98% range of the $c = 16$ accuracies, and the shift of the median accuracy (both are 1.000 and $+0.000$ at $c = 16$ by definition).

Dataset	Param.	32	48	64	96	128	192	256	384	512
CIFAR-10	SP	0.942/+0.031	0.889/+0.063	0.859/+0.079	0.818/+0.106	0.792/+0.112	0.728/+0.128	0.726/+0.136	0.654/+0.161	0.678/+0.154
CIFAR-10	μP	0.958/+0.022	0.921/+0.033	0.884/+0.042	0.892/+0.054	0.880/+0.053	0.863/+0.062	0.863/+0.060	0.869/+0.063	0.864/+0.062
SVHN	SP	0.928/+0.000	0.876/+0.048	0.846/+0.079	0.799/+0.113	0.761/+0.130	0.715/+0.159	0.704/+0.174	0.652/+0.224	0.700/+0.197
SVHN	μP	0.948/+0.000	0.911/+0.000	0.876/+0.000	0.880/+0.009	0.864/+0.000	0.850/+0.013	0.852/+0.011	0.855/+0.024	0.847/+0.002

Evaluation criteria for accuracy prediction. Because the prediction target drifts with width, one criterion alone cannot tell a metanetwork that has learned nothing about a new width from one that ranks the networks well but predicts values on the wrong scale. We therefore report four criteria for the CNN accuracy prediction task.

- The mean absolute error $\frac{1}{n_{\text{test}}} \sum_i |y_i - \hat{y}_i|$, for labels y_i and predictions $\hat{y}_i$ at a given width, is measured in units of accuracy and is not normalized.
- $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$, where $\bar{y}$ is the mean label at that width, compares the squared errors with the spread of the labels at that width: $R^2 = 1$ means the predictions match the labels exactly, $R^2 = 0$ means the predictor does no better than always guessing the mean label, and $R^2 < 0$ means it does worse than that constant baseline.
- R^2 after an optimal affine recalibration, $R^2_{\text{recal}} = \max_{a,b} R^2(y, a\hat{y} + b)$, allows the predictions to be shifted and rescaled first, so a drift that is affine costs nothing. Comparing it with R^2 tells apart the two ways R^2 can fail: low R^2 with high R^2_{recal} means the predictor gets the networks in the right relative positions but on the wrong scale, while both low means it fails outright.
- Kendall's τ_b (Kendall et al., 1938) is a rank correlation: it asks only whether the predicted ordering of the networks matches the true ordering of their accuracies, ignoring the values on either side, so any monotone drift leaves it unchanged. It is also the criterion best suited to model selection, where one only wants to pick the best network out of a pool and the predicted values themselves do not matter.

For the Kendall's τ_b criterion, we also report a baseline τ_{HP} that quantifies how much of the ranking at a test width is available without reading the wide weights at all: one may assume the best hyperparameters at the training width will also be the best hyperparameters at a test width, so one could simply rank the wide networks based on the hyperparameters that generated them. We quantify this by generating 200 networks per width that realizes the same hyperparameter draws at every width. Then, we compute the Kendall's τ_b between the accuracies of the same draw at the training width $c_0 = 16$ and at width c. The resulting τ_{HP} is included as a baseline in the τ_b panels of Figures 6 and 7 to indicate how much of the ranking is available without reading the wide weights at all.

Results. The main-text figure (Figure 2) shows the plain GMN in the forward direction. Figure 5 repeats it as Figure 5a alongside the other GMN variants (Figures 5b to 5d) for completeness. All four panels use the same four datasets, the same training widths, and the same test-width ranges.

For the CNN accuracy-prediction task, we further report the results forall four evaluation criteria side-by-side in Figures 6 and 7 for the CIFAR-10 and SVHN datasets respectively. We use the CIFAR-10 results as an example to show how these plots give a fine-grained understanding of the GMN's size generalization performance; similar patterns arise in the SVHN results, which we omit for brevity.

For example, compare the baseline ScaleGMN (forward) with its duplication-compatible counterpart under μP at width $c = 512$ (Figure 6c). The baseline ScaleGMN still achieves $\tau_b = 0.761$ but has $R^2 = -1.450$ with $R^2_{\text{recal}} = +0.682$: it still orders the wide networks well, and its predictions are merely on the wrong scale. Its duplication-compatible counterpart has an indistinguishable $\tau_b = 0.756$, so the modification does not improve the ranking, but its $R^2 = +0.714$ shows that it predicts the actual accuracy values much better. Because τ_b and R^2_{recal} barely change between the two

variants while R^2 swings by nearly 2, the gain is specifically in tracking the accuracy drift with width, not in ranking or in affine-corrected fit.

Under SP, both the baseline GMN (forward) and the duplication-compatible GMN (forward) perform badly on all four criteria (Figure 6a), indicating a genuine failure: neither the actual accuracy values nor the ranking of the networks is recovered.

By contrast, the matrix-product GMN and the matrix-product ScaleGMN under μP (Figures 6a and 6c) are where all four criteria agree that size generalization is reasonably good.

E THEORY OF SIZE GENERALIZATION ON INDEPENDENTLY TRAINED WEIGHTS

Distribution of trained weights. Mathematically, fix a training rule. The training procedure for the width-n network is driven by three sources of randomness:

- ξ_n: the random initialization of the width-n network;
- Γ: the hyperparameters (learning rate, weight decay, batch size, training horizon, etc.), with law λ;
- S: the remaining random choices, such as the minibatch sequence, data augmentations or dropout masks, and optimizer noise.

We assume Γ, S, ξ_n are mutually independent, and that λ does not depend on n.

For width n, define

$$\theta_n = \mathsf{Train}_n(\Gamma, S, \xi_n), \qquad \mu_n := \mathrm{Law}(\theta_n).$$

The fixed-hyperparameter case is recovered by taking $\lambda = \delta_\gamma$, so that $\mu_n = \mu_{n,\gamma}$.

Width-independent hyperparameter distribution. In practice, the best hyperparameters often vary with width and are usually tuned for each individual width n, giving distributions of the form $\mu_{n,\gamma_n(S)}$ rather than a single width-independent hyperparameter law. We cannot rigorously analyze $\mu_{n,\gamma_n(S)}$, since sweeping hyperparameters adds a further layer of complication, so we later rely only on numerical visualizations for this case. When the law of Γ is instead fixed across widths n, the distribution (μ_n) is tractable under the same uniform or integrable regularity assumptions on the hyperparameters, and can be fully characterized rigorously using existing infinite-width training-dynamics analysis — in particular, the toolkit of Tensor Programs.

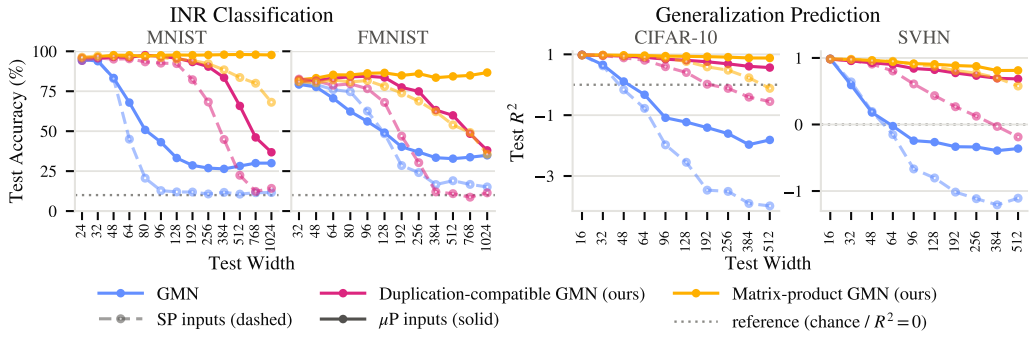
This width-independent-hyperparameter assumption is a genuine restriction. First, it fixes the training horizon, or its law, independently of n. Second, it means the theoretical results are most relevant to practice in the μP setting, under which the optimal hyperparameters are widely observed to be width-independent: γ need not be tuned separately for each n, since even when tuned, γ_n is roughly constant across n. Size generalization below is therefore most meaningful for input weights trained under μP.

E.1 PRELIMINARY: TENSOR PROGRAMS AND μP

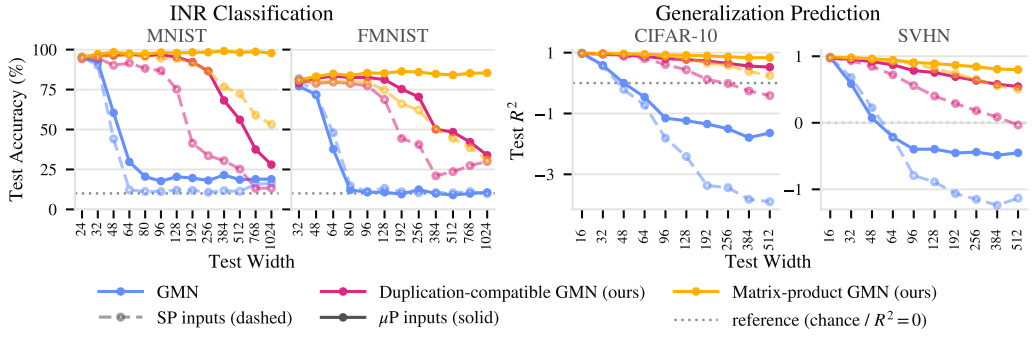
Tensor programs. We briefly review the Tensor Programs framework developed in Littwin & Yang (2023). This framework gives a rigorous way to characterize infinite-width limits for a broad class of neural-network computations, including training dynamics of all standard architectures using standard entry-wise optimizers.

The framework applies to the following setting: The dataset is fixed, and the number of training steps T is fixed independently of the width n. Randomness enters only through the initialization and then propagates to every downstream quantity, including activations, gradients, and weight updates. We are interested in the limiting behavior of these quantities (as random variables) as the width $n \to \infty$.

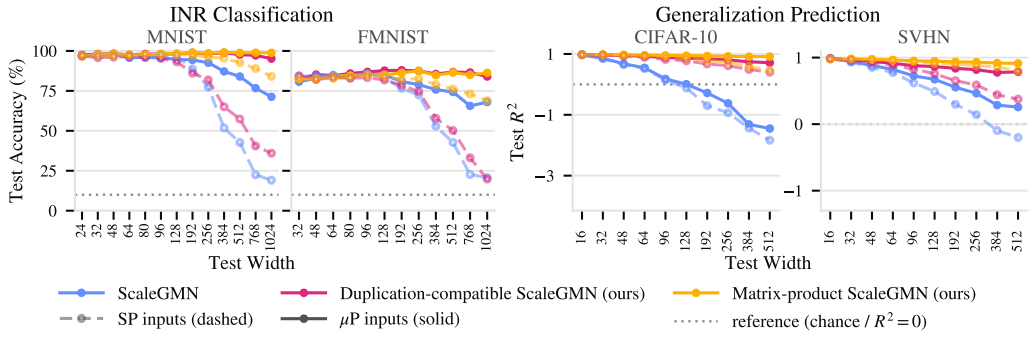
The high-level idea is the following. The basic object is a *tensor program*, an abstract programming language for computations built from random initialization, matrix multiplication, averaging over the width coordinate, and entrywise nonlinearities. See Definition 13 below. The Tensor Programs



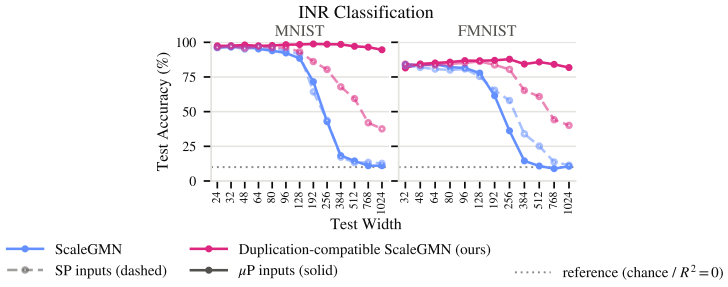
(a) GMN (forward).



(b) GMN (bidirectional).

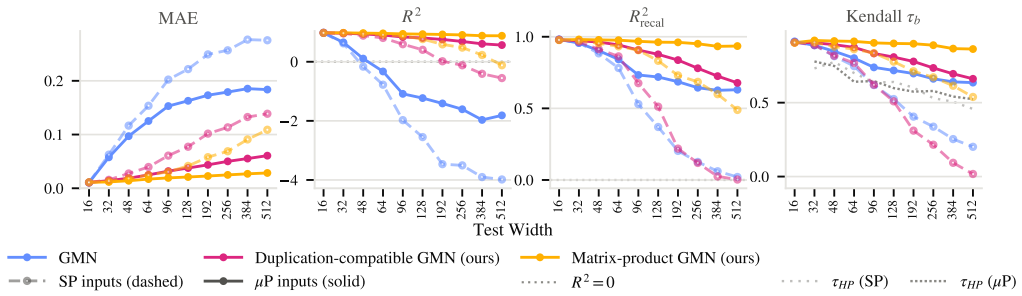


(c) ScaleGMN (forward).

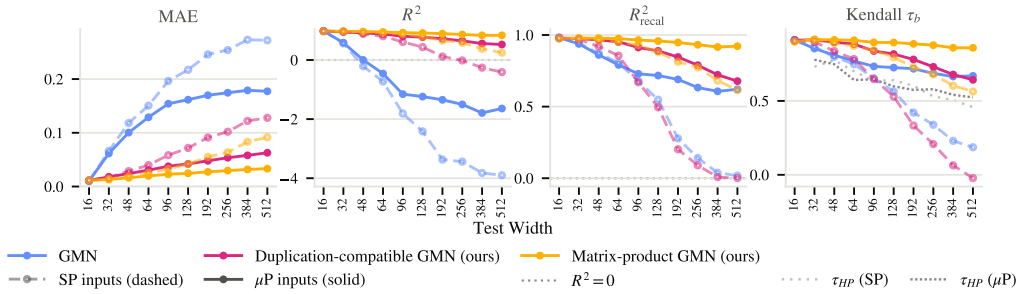


(d) ScaleGMN (bidirectional). (Bidirectional ScaleGMN does not have a valid matrix-product variant; Also, bidirectional ScaleGMN is not evaluated on the CNN tasks because of training instability.)

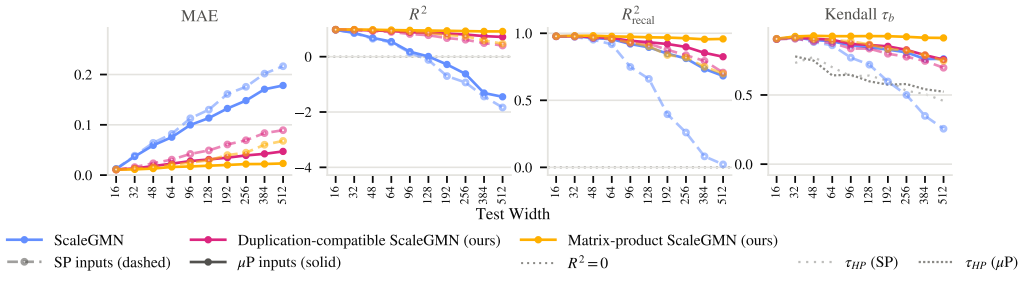
Figure 5: Size generalization on wider input networks trained independently from random initialization, under either μP or SP. Figure 5b repeats the main-text figure (Figure 2) for completeness.



(a) GMN (forward).

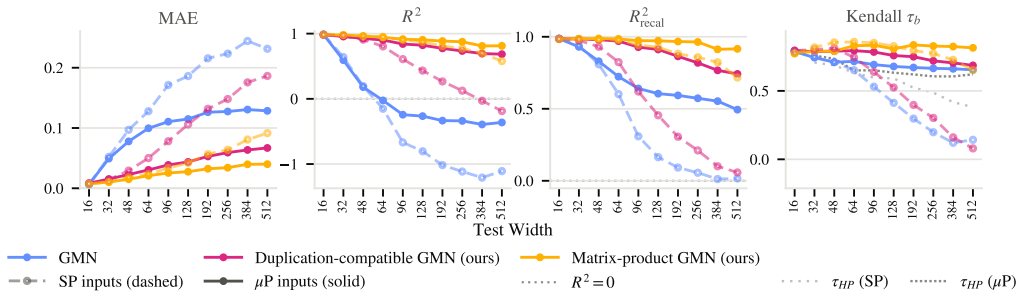


(b) GMN (bidirectional).

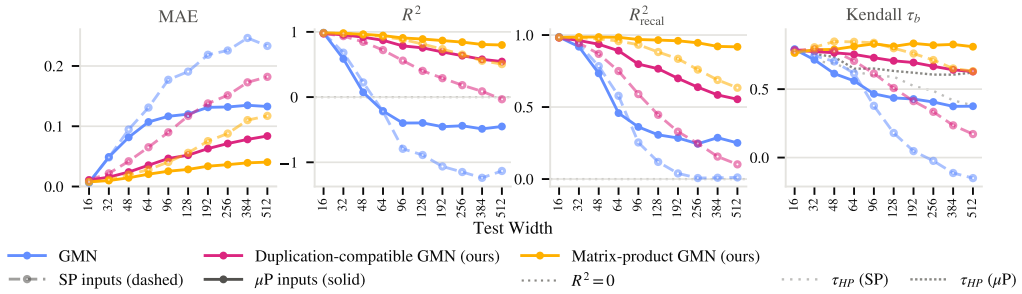


(c) ScaleGMN (forward).

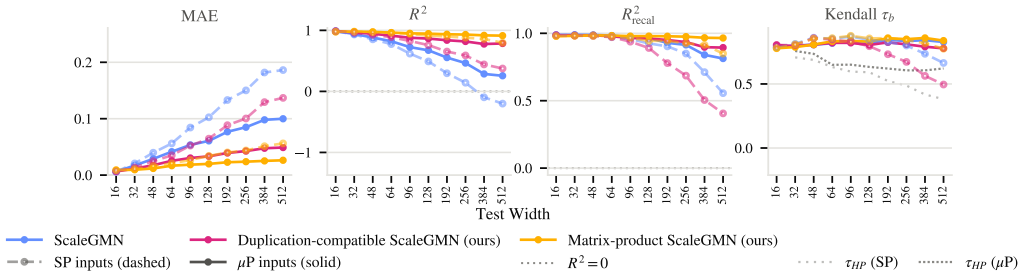
Figure 6: The four evaluation criteria for accuracy prediction on the CIFAR-10 CNN zoo, one row of panels per setting.



(a) GMN (forward).



(b) GMN (bidirectional).



(c) ScaleGMN (forward).

Figure 7: The four evaluation criteria for accuracy prediction on the SVHN CNN zoo, one row of panels per setting.

papers show, first, that many neural computations used in practice, including convolutions, attention, batch and layer normalizations, and residual connections, can be expressed as tensor programs; see Appendix A of Yang (2019). They also show that if an architecture's forward pass can be written as a tensor program, then its backward pass and a fixed finite number of weight updates can be written as a tensor program; see Section 2.9.3 of Littwin & Yang (2023). Thus, to understand finite-time infinite-width training dynamics, it suffices to characterize the infinite-width limits of tensor programs. For this purpose, one assigns to each tensor program a recursively defined collection of random variables and deterministic scalars that describe the limiting objects; see Definition 2.6.5 of Littwin & Yang (2023). The master theorem then states the precise mode in which the finite-width program converges to these limits.

We now give the concrete form of the tensor programs that we need.

Definition 13 (Tensor program, Definition 2.6.1 of Littwin & Yang (2023))**.** A *tensor program* of width n generates a sequence $\boldsymbol{x}$ of vectors in $\mathbb{R}^n$ and a sequence $\boldsymbol{c}$ of scalars in $\mathbb{R}$, starting from an initial set $\boldsymbol{c}^0 \subseteq \boldsymbol{c}$ of random scalars, an initial set $\boldsymbol{x}^0 \subseteq \boldsymbol{x}$ of random vectors in $\mathbb{R}^n$, and an initial set $\mathcal{W}$ of random $n \times n$ matrices. At each step, one of the following three instructions is applied.

AVG Choose a vector $x \in \boldsymbol{x}$ and append to $\boldsymbol{c}$ the scalar

$$\frac{1}{n}\sum_{\alpha=1}^{n} x_\alpha \in \mathbb{R}.$$

MATMUL Choose a matrix $W \in \mathcal{W}$ and a vector $x \in \boldsymbol{x}$, and append to $\boldsymbol{x}$ the vector

$$Wx \in \mathbb{R}^n \quad \text{or} \quad W^\top x \in \mathbb{R}^n.$$

OUTERNONLIN Choose an integer $r \geq 0$ and a function $\psi\colon \mathbb{R}^{|\boldsymbol{x}|(r+1)+|\boldsymbol{c}|} \to \mathbb{R}$, and append to $\boldsymbol{x}$ the vector $y \in \mathbb{R}^n$ with

$$y_\alpha = \frac{1}{n^r}\sum_{\beta_1,\ldots,\beta_r=1}^{n} \psi(\boldsymbol{x}_\alpha; \boldsymbol{x}_{\beta_1}; \ldots; \boldsymbol{x}_{\beta_r}; \boldsymbol{c}),$$

where $\boldsymbol{x}_\gamma \in \mathbb{R}^{|\boldsymbol{x}|}$ is the γth row of $\boldsymbol{x}$, viewed as the $n \times |\boldsymbol{x}|$ matrix whose columns are the vectors in $\boldsymbol{x}$.

The vector variables record quantities with one dimension that scales with the width n, such as pre-activations, activations, and their gradients. The scalar variables record quantities whose dimension does not scale with the width, such as network outputs and losses.

Every vector in $\boldsymbol{x}$ has an associated limiting random variable, and every scalar $\theta \in \boldsymbol{c}$ has a deterministic limit $\mathring{\theta}$. Both are defined by the recursion of Definition 2.6.5 of Littwin & Yang (2023). We now state the master theorem of Littwin & Yang (2023) that we will use later for our analysis, which characterizes the precise sense in which the scalars $\boldsymbol{c}$ of a tensor program converge to their infinite-width limits $\mathring{\boldsymbol{c}}$ as $n \to \infty$. This first requires some mild regularity assumptions on the random initialization and the nonlinearities used in the tensor program.

Assumption 1 (Setup 2.6.3 of Littwin & Yang (2023))**.** In the tensor program of Definition 13,

(a) every entry of every initial matrix $W \in \mathcal{W}$ is i.i.d. $\mathcal{N}(0, 1/n)$;

(b) every entry of every initial vector in $\boldsymbol{x}^0$ is i.i.d. $\mathcal{N}(0, 1)$;

(c) the initial scalars satisfy $\boldsymbol{c}^0 \xrightarrow{\text{a.s.}} 0$ as $n \to \infty$; and

(d) every $\psi\colon \mathbb{R}^k \to \mathbb{R}$ used in OUTERNONLIN is *pseudo-Lipschitz*: there are constants $C, d < \infty$ such that

$$|\psi(x) - \psi(y)| \leq C\|x - y\|_2 \left(1 + \sum_{i=1}^{k}|x_i|^d + \sum_{i=1}^{k}|y_i|^d\right) \qquad \text{for all } x, y \in \mathbb{R}^k.$$

Theorem 14 (Master theorem, Theorem 2.6.10 of Littwin & Yang (2023))**.** *Let $\boldsymbol{c} = \boldsymbol{c}(n)$ be any finite collection of scalars produced by a tensor program satisfying Assumption 1, with the size of*

this collection independent of n. Let $\mathring{\boldsymbol{c}}$ be the deterministic vector assigned to these scalars by Definition 2.6.5 of Littwin & Yang (2023). Then, as $n \to \infty$,

$$\boldsymbol{c} \xrightarrow{\text{a.s.}} \mathring{\boldsymbol{c}}.$$

We now state a corollary of the master theorem that shows empirical measures of vectors produced by a tensor program converge in Wasserstein distance to the law of their limiting random variables. This puts the master theorem in a form that is more convenient for our later analysis of size generalization.

Corollary 15. *Consider a tensor program satisfying Assumption 1. Let $v_n^1, \dots, v_n^K \in \mathbb{R}^n$ be any fixed finite collection of vectors produced by the program, with associated limiting random variables $V^1, \dots, V^K$. Then the following hold.*

(i) For every pseudo-Lipschitz $\psi\colon \mathbb{R}^K \to \mathbb{R}$,

$$\frac{1}{n}\sum_{i=1}^{n} \psi\left(v_{n,i}^1, \dots, v_{n,i}^K\right) \xrightarrow{\text{a.s.}} \mathbb{E}\left[\psi(V^1, \dots, V^K)\right] \in \mathbb{R}.$$

(ii) Define $\xi_{n,i} := (v_{n,i}^1, \dots, v_{n,i}^K)$ with empirical measure $\widehat{\nu}_n := \frac{1}{n}\sum_{i=1}^n \delta_{\xi_{n,i}}$, and let $\nu_\infty := \mathrm{Law}(V^1, \dots, V^K)$. Then for every $p \geq 1$,

$$W_p(\widehat{\nu}_n, \nu_\infty) \to 0 \qquad \textit{almost surely}.$$

Proof. For part (i), extend the program by applying OUTERNONLIN with $r = 0$ to the selected vectors. This produces the vector with entries $\psi(v_{n,i}^1, \dots, v_{n,i}^K)$. By the OUTERNONLIN rule in Definition 2.6.5 of Littwin & Yang (2023), its limiting random variable is $\psi(V^1, \dots, V^K)$. Applying AVG to this vector produces the scalar $\frac{1}{n}\sum_{i=1}^n \psi\left(v_{n,i}^1, \dots, v_{n,i}^K\right)$, and the AVG rule assigns to it the deterministic limit $\mathbb{E}\,\psi(V^1, \dots, V^K)$. Since ψ is pseudo-Lipschitz, the extended program still satisfies Assumption 1. Theorem 14 therefore applies to this scalar and gives the desired convergence.

For part (ii), part (i) implies that for every pseudo-Lipschitz $\psi\colon \mathbb{R}^K \to \mathbb{R}$,

$$\int \psi \,\mathrm{d}\widehat{\nu}_n = \frac{1}{n}\sum_{i=1}^{n} \psi(\xi_{n,i}) \xrightarrow{\text{a.s.}} \mathbb{E}\,\psi(V^1, \dots, V^K) = \int \psi \,\mathrm{d}\nu_\infty, \tag{43}$$

with finite limit. Take a countable pseudo-Lipschitz convergence-determining class $\{\phi_j\}_{j\geq 1}$ for weak convergence of probability measures on $\mathbb{R}^K$; for example, the proof of Proposition 4.4 in Ethier & Kurtz (2009) gives an explicit construction. For every fixed j, the function ϕ_j is pseudo-Lipschitz, so (43) holds with $\psi = \phi_j$ on an event of probability one. Intersecting these countably many events gives an event of probability one on which

$$\int \phi_j \,\mathrm{d}\widehat{\nu}_n \to \int \phi_j \,\mathrm{d}\nu_\infty \qquad \text{for every } j.$$

Therefore, $\widehat{\nu}_n \Rightarrow \nu_\infty$ almost surely.

We now upgrade this weak convergence to Wasserstein convergence. For each positive integer q, the function $\psi_q(x) = \|x\|_2^q$ is pseudo-Lipschitz. Applying (43) to these countably many functions and intersecting the corresponding probability-one events gives

$$\int \|x\|_2^q \,\mathrm{d}\widehat{\nu}_n(x) \to \int \|x\|_2^q \,\mathrm{d}\nu_\infty(x) \qquad \text{for every positive integer } q$$

almost surely. On the intersection with the weak-convergence event, Theorem 6.9 of Villani et al. (2009) gives $W_q(\widehat{\nu}_n, \nu_\infty) \to 0$ for every positive integer q.

Finally, given any real number $p \geq 1$, choose an integer $q \geq p$. Then, $W_p(\widehat{\nu}_n, \nu_\infty) \leq W_q(\widehat{\nu}_n, \nu_\infty) \to 0$ almost surely. □

Beyond almost sure convergence, we now prove uniform moment bounds so that later this can be upgraded to convergence in L^p. Notice that this lemma focuses on the case of matrix-free tensor programs, which is sufficient for our purposes later. For general tensor programs, this requires stronger assumptions on the nonlinearities to be polynomially bounded (see e.g. Proposition 4.3.4 of Littwin & Yang (2023)).

Lemma 16 (Uniform moment bound for matrix-free tensor programs)**.** *Consider a tensor program satisfying Assumption 1, and suppose that its matrix set $\mathcal{W}$ is empty. Suppose additionally that the initial scalars have moments of all orders bounded uniformly in n. Then, for every positive integer p, every vector $x = x(n)$ produced by the program, and every scalar $c = c(n)$ produced by the program,*

$$\sup_n \mathbb{E}\left[\frac{1}{n}\sum_{\alpha=1}^{n}|x_\alpha|^p\right] < \infty \qquad \textit{and} \qquad \sup_n \mathbb{E}\,|c|^p < \infty.$$

Proof. We argue by induction over the finite sequence of program instructions, proving the two bounds simultaneously for every positive integer p.

Base case. They hold for the initial vectors because their entries are standard Gaussian, and they hold for the initial scalars by the additional moment assumption.

Induction step. Consider first that an AVG instruction produces $c = \frac{1}{n}\sum_{\alpha=1}^{n} x_\alpha$. Since the function $u \mapsto |u|^p$ is convex, Jensen's inequality gives $\mathbb{E}\,|c|^p \le \mathbb{E}\left[\frac{1}{n}\sum_{\alpha=1}^{n}|x_\alpha|^p\right]$, which is bounded uniformly in n by the induction hypothesis.

Consider next that an OUTERNONLIN instruction produces

$$y_\alpha = \frac{1}{n^r}\sum_{\beta_1,\ldots,\beta_r=1}^{n} \psi(\boldsymbol{x}_\alpha; \boldsymbol{x}_{\beta_1}; \ldots; \boldsymbol{x}_{\beta_r}; \boldsymbol{c}).$$

Since ψ is pseudo-Lipschitz (Assumption 1), the pseudo-Lipschitz inequality with $y = 0$ gives, for every $z \in \mathbb{R}^k$, $|\psi(z)| \le |\psi(0)| + C\|z\|_2\left(1 + \sum_{i=1}^{k}|z_i|^d\right)$. It follows that, for each positive integer p, there are a constant $C_p < \infty$ and a positive integer q_p such that for every $z \in \mathbb{R}^k$, $|\psi(z)|^p \le C_p\left(1 + \sum_{j=1}^{k}|z_j|^{q_p}\right)$. Applying Jensen's inequality yields

$$\mathbb{E}\left[\frac{1}{n}\sum_{\alpha=1}^{n}|y_\alpha|^p\right] \le \frac{1}{n^{r+1}}\sum_{\alpha,\beta_1,\ldots,\beta_r=1}^{n}\mathbb{E}\,|\psi(\boldsymbol{x}_\alpha; \boldsymbol{x}_{\beta_1}; \ldots; \boldsymbol{x}_{\beta_r}; \boldsymbol{c})|^p.$$

Write the previously produced vectors and scalars that enter this instruction as $\boldsymbol{x} = (x^1, \ldots, x^m)$ and $\boldsymbol{c} = (c^1, \ldots, c^s)$. The polynomial envelope then gives a constant $C'_p < \infty$, independent of n, such that the last display is at most

$$C'_p\left(1 + \sum_{a=1}^{m}\mathbb{E}\left[\frac{1}{n}\sum_{\gamma=1}^{n}\left|x^a_\gamma\right|^{q_p}\right] + \sum_{b=1}^{s}\mathbb{E}\left|c^b\right|^{q_p}\right).$$

The induction hypothesis bounds each of these finitely many quantities uniformly in n.

These are all possible instructions because $\mathcal{W}$ is empty, so the induction proves the claim. □

μP and hyperparameter transfer. Using the Tensor Programs framework, Littwin & Yang (2023) proves that μP is the unique parameterization in a large class of width-dependent parameterizations that satisfies a range of desirable properties. In this sense, it is the "optimal" parameterization in the infinite-width limit. Table 7 summarizes the concrete scalings for SP and μP.

Table 7: Initialization variance and example hyperparameters for different MLP parameters under SP and μP trained with SGD and Adam (Yang et al., 2021). μP is also defined for any standard architecture and prescribe scalings for weight-decay coefficients. See Littwin & Yang (2023) for details. In both cases, the output bias $b^{(L+1)}$ is initialized to zero with constant learning rate. For a weight matrix W taking inputs x as Wx, fan_in and fan_out are the number of columns and rows respectively.

	Init variance		SGD learning rate		AdamW learning rate	
Input weights $W^{(1)}$; Biases $(b^{(\ell)})_{\ell=1}^{L}$	1/fan_in	1/fan_in	1	fan_out	1	1
Hidden weights $(W^{(\ell)})_{\ell=2}^{L}$	1/fan_in	1/fan_in	1	1	1	1/fan_in
Output weights $W^{(L+1)}$	1/fan_in	1/fan_in^2	1	1/fan_in	1	1/fan_in

Furthermore, Yang et al. (2021) empirically observed that μP allows hyperparameters tuned at a small width to be transferred to larger widths without retuning, a phenomenon called *hyperparameter transfer*. This is shown in Figure 8, where the optimal learning rate is stable across a wide range of widths under μP, but moves considerably under SP. This is precisely because under μP the training dynamics of the networks of different widths are similar to each other, as they converge to the same well-behaved limit with controlled finite approximation error.

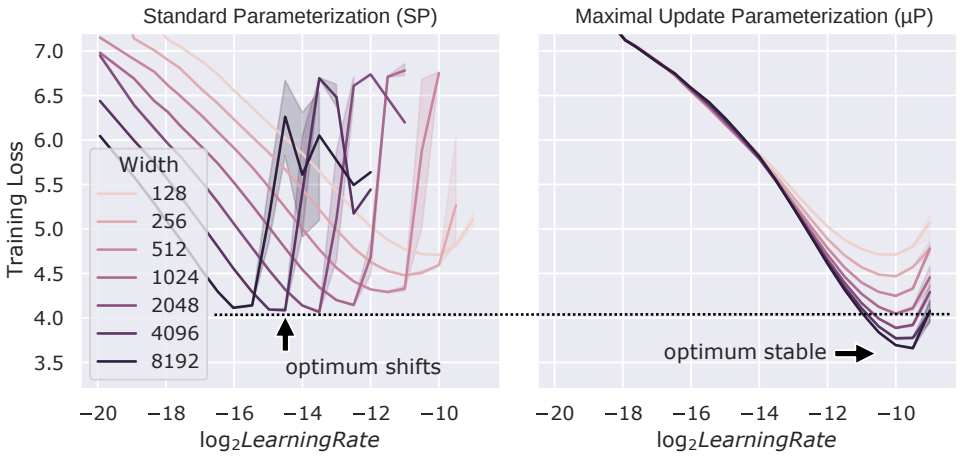


Figure 8: Hyperparameter transfer under μP, Figure 1 of Yang et al. (2021). Training loss against learning rate for Transformers of width ranging over $128, \ldots, 8192$, trained with Adam.

E.2 Proof of Theorem 4

Throughout this subsection the input MLP has $L = 1$ layer, its input and output dimensions $d_{\text{in}}, d_{\text{out}}$ are fixed, and its single hidden width is n, so its forward pass is

$$f_n^{\theta}(x) = W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)},$$

with $W^{(1)} \in \mathbb{R}^{n\times d_{\text{in}}}$, $b^{(1)} \in \mathbb{R}^n$, $W^{(2)} \in \mathbb{R}^{d_{\text{out}}\times n}$, and $b^{(2)} \in \mathbb{R}^{d_{\text{out}}}$. Recall that the fan-in rescaled coordinates are $\overline{W}^{(1)} = d_{\text{in}}W^{(1)}$ and $\overline{W}^{(2)} = nW^{(2)}$.

In this 1-layer MLP setting, μP is exactly the classical mean-field scaling. Hence the results in this part could be proved directly using mean-field limit results such as (Mei et al., 2018; Chizat & Bach, 2018) instead of the Tensor Programs framework. We nevertheless use the Tensor Programs formulation below to keep the proof strategy consistent with the later size-generalization result, where the same machinery is needed.

We first give a high-level explanation in Appendix E.2.1, where we also compare μP with SP and explain why the parameterization matters. It is followed by the detailed proof in Appendix E.2.2.

E.2.1 A high-level explanation

We first explain intuitively why we expect $W_1(\mu_n, \mu_m; d_\infty^{\text{op}}) \to 0$ if the training of the networks is done under μP, and why this is not expected to hold under SP.

What μP prescribes, and why it makes the widths agree. What μP (Table 7) prescribes for the 1-layer MLP is summarized below:

μP	$W^{(1)}$	$b^{(1)}$	$W^{(2)}$	$b^{(2)}$
initial variance	$1/d_{\text{in}}$	$1/d_{\text{in}}$	$1/n^2$	0
SGD rate	n	n	$1/n$	1
AdamW rate	1	1	$1/n$	1

At initialization, $\overline{W}^{(1)} = d_{\text{in}} W^{(1)}$, $b^{(1)}$, and $\overline{W}^{(2)} = n W^{(2)}$ all have i.i.d. entries with distributions independent of n. Grouping the entries attached to one and the same hidden neuron, the n vectors

$$\zeta_{n,i}(\theta) := \left(\frac{\overline{W}^{(1)}_{i,:}}{\sqrt{d_{\text{in}}}},\, b^{(1)}_i,\, \frac{\overline{W}^{(2)}_{:,i}}{\sqrt{d_{\text{out}}}} \right) \in \mathbb{R}^{d_{\text{in}}+1+d_{\text{out}}} \qquad (i \in [n]) \tag{44}$$

are i.i.d. draws from one Gaussian law that does not depend on n. Meanwhile, the output bias $b^{(2)} \in \mathbb{R}^{d_{\text{out}}}$ is initialized to zero. A freshly initialized width-n network is therefore nothing more than n independent samples from one fixed law, together with a single d_{out}-dimensional vector. A wider network draws more samples from the same law; it does not change the law. This is the sense in which the different widths are "the same network seen at different resolutions".

The learning rate scalings prescribed by μP are exactly what is needed to preserve this picture through training, rather than only at initialization. At every fixed step t, the n vectors $\zeta_{n,i}(\theta_n(t))$ attached to the hidden neurons $i \in [n]$ still "behave like" i.i.d. draws from a law that depends on t but not on n. The precise sense of "behave like" is that the empirical distribution of the n vectors converges almost surely as $n \to \infty$: there exists some limiting distribution $\nu_\infty(t)$ on $\mathbb{R}^{d_{\text{in}}+1+d_{\text{out}}}$ such that

$$W_2\left(\frac{1}{n} \sum_{i=1}^{n} \delta_{\zeta_{n,i}(\theta_n(t))},\, \nu_\infty(t) \right) \xrightarrow{\text{a.s.}} 0$$

as $n \to \infty$, where W_2 is the Wasserstein-2 distance. Moreover, the behavior of the output bias during the training is also predictable: there exists some limiting $b^{(2)}_\infty(t) \in \mathbb{R}^{d_{\text{out}}}$ such that $b^{(2)}_n(t) \xrightarrow{\text{a.s.}} b^{(2)}_\infty(t)$ as $n \to \infty$. The Tensor Programs framework allows us to precisely describe the limiting law $\nu_\infty(t)$ and the limiting bias $b^{(2)}_\infty(t)$ at every step t, and Corollary 15 gives the rigorous proof of the convergence.

It remains to relate this convergence to the closeness we actually want, namely closeness of two trained networks of different widths n and m in d^{op}_∞, the symmetrized distance induced by the normalized spectral norm. It turns out that in the 1-layer MLP case the distance in d^{op}_∞ is controlled by the Wasserstein-2 distance between the two empirical distributions of the vectors $\zeta_{n,i}$ together with the difference between the two output biases. Both of these are small, since by the above each of the two empirical distributions is close to the common $\nu_\infty(t)$ and each output bias is close to the common $b^{(2)}_\infty(t)$. Hence the trained networks of widths n and m are close in d^{op}_∞, which in turn gives the desired closeness of μ_n and μ_m in $W_1(\cdot; d^{\text{op}}_\infty)$.

What SP prescribes, and why the widths disagree already at initialization. The same specialization of Table 7 gives

SP	$W^{(1)}$	$b^{(1)}$	$W^{(2)}$	$b^{(2)}$
initial variance	$1/d_{\text{in}}$	$1/d_{\text{in}}$	$1/n$	0
SGD rate	1	1	1	1
AdamW rate	1	1	1	1

At $L = 1$ the two parameterizations thus differ only in the output layer, whose initial variance is $1/n$ rather than $1/n^2$ and whose rate is 1 rather than $1/n$, together with the input-layer SGD rate.

Let $\mu_n(0) := \text{Law}(\theta_n(0))$ denote the initialization law at width n. Already at initialization, the sequence $(\mu_n(0))_n$ is not Cauchy in $W_1(\,\cdot\,, \cdot\,; d^{\text{op}}_\infty)$ under SP. With d_{out} fixed, the standard random matrix estimate for the largest singular value gives $\mathbb{E}\left\|W^{(2)}(0)\right\| \to 1$, and in particular $\mathbb{E}\left\|W^{(2)}(0)\right\| = \Theta(1)$. As such,

$$\mathbb{E}\left\|\theta_n(0)\right\|_{V_n,\text{op}} = \sqrt{\frac{n}{d_{\text{out}}}}\, \mathbb{E}\left\|W^{(2)}(0)\right\| + O(1) = \Theta(\sqrt{n}),$$

because the first-layer and bias contributions are $O(1)$. In other words, the initialized weights of different widths are not expected to be close, as they are farther and farther away from the origin in the normalized spectral norm as $n \to \infty$.

As a result, for every coupling π of $\mu_n(0)$ and $\mu_{2n}(0)$, if $(\theta_n(0), \theta_{2n}(0))$ denotes the pair sampled from π, then the reverse triangle inequality gives

$$\mathop{\mathbb{E}}_{\pi} d_\infty^{\mathrm{op}}(\theta_n(0), \theta_{2n}(0)) \geq \left| \mathbb{E} \left\|\theta_{2n}(0)\right\|_{V_{2n},\mathrm{op}} - \mathbb{E} \left\|\theta_n(0)\right\|_{V_n,\mathrm{op}} \right| = \Theta(\sqrt{n}),$$

where we used that $d_\infty^{\mathrm{op}}(\theta, 0) = \|\theta\|_{V_r,\mathrm{op}}$ for $\theta \in V_r$. Hence,

$$W_1\left(\mu_n(0), \mu_{2n}(0); d_\infty^{\mathrm{op}}\right) \to \infty.$$

The same initialization argument applies at every depth $L \geq 1$, showing that SP-initialized weights are generally far apart in d_∞^{op}. The training process is unlikely to remedy this, as we numerically checked in various experiments.

E.2.2 Proofs of the 1-layer results

We first state the additional regularity assumptions on the 1-layer MLP model and its training. These assumptions ensure that the entire training process is a tensor program satisfying Assumption 1, so that Theorem 14 and corollary 15 apply.

Assumption 2. (i) The weights and biases are initialized with i.i.d. Gaussian entries.

(ii) Conditional on every $(\Gamma, S) = (\gamma, s)$ in the support of (Γ, S), the training horizon $T = T(\gamma)$ is finite and independent of the width.

(iii) The optimizer is an entrywise optimizer with update function $Q_t \colon \mathbb{R}^{t+1} \to \mathbb{R}$ as defined in Section 2.1 of Littwin & Yang (2023), which includes SGD and Adam as special cases.

(iv) The activation σ, its derivative σ', the derivative of the training loss, and the optimizer maps Q_t for $t < T$ are pseudo-Lipschitz in the sense of Assumption 1(d).

These assumptions are inherited from the Tensor Programs framework. They are for technical convenience rather than a fundamental restriction, and are likely to be removed or relaxed as noted in Littwin & Yang (2023).

Proposition 17 (Trained weights at large widths agree in the normalized spectral distance)**.** *Assume that Assumption 2 holds. Suppose that there exists $q > 1$ such that*

$$\sup_{(\gamma,s)\in \mathrm{supp}(\Gamma,S)} \sup_{n\geq 1} \mathop{\mathbb{E}}_{\xi_n} \|\theta_n\|_{V_n,\mathrm{F}}^q < \infty.$$

Then, for every $p \in [1, q)$,

$$\sup_{m\geq n} W_p(\mu_n, \mu_m; d_\infty^{\mathrm{op}}) \longrightarrow 0 \qquad \text{as } n \to \infty. \tag{45}$$

Proof. We first fix (γ, s) in the support of (Γ, S) and condition on $(\Gamma, S) = (\gamma, s)$. This fixes the hyperparameters γ and the auxiliary training randomness s, so that the initialization is the only remaining source of randomness. Throughout the conditional argument, θ_n denotes $\mathsf{Train}_n(\gamma, s, \xi_n)$, and we realize $(\xi_n)_{n\geq 1}$ as an independent family on a common product probability space, and interpret all almost-sure statements with respect to this coupling.

The run as a tensor program. Under Assumption 2, training the 1-layer MLP is a tensor program satisfying Assumption 1. See Section 2.1 of Yang & Hu (2021) for the construction for the bias-free case trained with SGD. Notice that this tensor program requires no random $n \times n$ matrix. The μP scalings ensure that $\overline{W}^{(1)}$, $b^{(1)}$, and $\overline{W}^{(2)}$ have entries of order one throughout every fixed finite training run, and Definition 2.6.5 of Littwin & Yang (2023) assigns limiting random variables to these entries.

Append to the tensor program the vectors $\overline{W}^{(1)}_{:,a}(T)/\sqrt{d_{\mathrm{in}}}$ for $a \in [d_{\mathrm{in}}]$ and $\overline{W}^{(2)}_{b,:}(T)/\sqrt{d_{\mathrm{out}}}$ for $b \in [d_{\mathrm{out}}]$. Each is produced from an existing vector by a linear OuterNonlin instruction, so the

extended program still satisfies Assumption 1. Together with $b^{(1)}(T)$, these vectors have the joint coordinates $\zeta_{n,i}(\theta_n(T))$ defined in (44). Applying Corollary 15(ii) with $K = d_{\text{in}} + 1 + d_{\text{out}}$ gives

$$W_2\left(\frac{1}{n}\sum_{i=1}^{n}\delta_{\zeta_{n,i}(\theta_n(T))}, \nu_\infty(T)\right) \xrightarrow{\text{a.s.}} 0,$$

where $\nu_\infty(T)$ is the conditional limiting probability measure on $\mathbb{R}^{d_{\text{in}}+1+d_{\text{out}}}$. The entries of $b^{(2)}(T)$ are scalars of the program, so Theorem 14 gives $b_n^{(2)}(T) \xrightarrow{\text{a.s.}} b_\infty^{(2)}(T)$ for a limiting output bias $b_\infty^{(2)}(T) \in \mathbb{R}^{d_{\text{out}}}$.

Almost-sure convergence in d_∞^{F}, and hence in d_∞^{op}. The width-n weight space is $V_n := \mathbb{R}^{n\times d_{\text{in}}} \oplus \mathbb{R}^n \oplus \mathbb{R}^{d_{\text{out}}\times n} \oplus \mathbb{R}^{d_{\text{out}}}$. Since the hidden-neuron permutations act on all the n-dimensional coordinates simultaneously, it is equivalent to regard V_n as $\mathbb{R}^{n(d_{\text{in}}+1+d_{\text{out}})} \oplus \mathbb{R}^{d_{\text{out}}}$. Recall that the normalized Frobenius norm (14) at $L = 1$ reads

$$\|\theta\|_{V_n,\text{F}}^2 = \frac{1}{n}\|\zeta_n(\theta)\|_{\text{F}}^2 + \frac{1}{d_{\text{out}}}\|b^{(2)}\|_2^2.$$

The corresponding symmetrized distance d_∞^{F} on the limit space coincides with the Wasserstein-2 distance between the empirical measures of the stacked hidden-neuron features, combined in ℓ_2 with the output-bias distance (Appendix E.1 of Levin et al. (2026)): for $\theta \in V_n$ and $\widetilde{\theta} \in V_m$,

$$\left(d_\infty^{\text{F}}(\theta,\widetilde{\theta})\right)^2 = W_2^2\left(\frac{1}{n}\sum_{i=1}^{n}\delta_{\zeta_{n,i}(\theta)}, \frac{1}{m}\sum_{i=1}^{m}\delta_{\zeta_{m,i}(\widetilde{\theta})}\right) + \frac{1}{d_{\text{out}}}\|b^{(2)} - \widetilde{b}^{(2)}\|_2^2,$$

where W_2 is computed on $\mathbb{R}^{d_{\text{in}}+1+d_{\text{out}}}$ with ground metric $d_\zeta(u,v) = \|u-v\|_2$.

Combining with the Tensor Programs convergence results above, we have

$$\begin{aligned} d_\infty^{\text{F}}(\theta_n(T),\theta_m(T)) \le{}& W_2\left(\frac{1}{n}\sum_{i=1}^{n}\delta_{\zeta_{n,i}(\theta_n(T))}, \nu_\infty(T)\right) + W_2\left(\frac{1}{m}\sum_{i=1}^{m}\delta_{\zeta_{m,i}(\theta_m(T))}, \nu_\infty(T)\right) \\ &+ \frac{1}{\sqrt{d_{\text{out}}}}\|b_n^{(2)}(T) - b_\infty^{(2)}(T)\|_2 + \frac{1}{\sqrt{d_{\text{out}}}}\|b_m^{(2)}(T) - b_\infty^{(2)}(T)\|_2 \xrightarrow{\text{a.s.}} 0 \end{aligned}$$

as $\min\{n,m\} \to \infty$. Then, by Proposition 9 which relates d_∞^{op} and d_∞^{F}, we have

$$d_\infty^{\text{op}}(\theta_n(T),\theta_m(T)) \le 2\, d_\infty^{\text{F}}(\theta_n(T),\theta_m(T)) \xrightarrow{\text{a.s.}} 0.$$

Convergence in L^p. Again by Proposition 9 which relates d_∞^{op} and d_∞^{F}, we have

$$d_\infty^{\text{op}}(\theta_n(T),\theta_m(T)) \le 2\, d_\infty^{\text{F}}(\theta_n(T),\theta_m(T)) \le 2\left(\|\theta_n(T)\|_{V_n,\text{F}} + \|\theta_m(T)\|_{V_m,\text{F}}\right).$$

Raising this to the qth power and using the bound $(a+b)^q \le 2^{q-1}(a^q+b^q)$ for $a,b \ge 0$ and $q > 1$, we have

$$\sup_{(\gamma,s)\in\text{supp}(\Gamma,S)} \sup_{n,m\ge 1} \mathop{\mathbb{E}}_{\xi} d_\infty^{\text{op}}(\theta_n(T),\theta_m(T))^q \le 2^{2q} \sup_{(\gamma,s)\in\text{supp}(\Gamma,S)} \sup_{n\ge 1} \mathop{\mathbb{E}}_{\xi_n} \|\theta_n\|_{V_n,\text{F}}^q < \infty. \tag{46}$$

Fix $p \in [1,q)$. The preceding display and the de la Vallée–Poussin criterion show that the family

$$\{d_\infty^{\text{op}}(\theta_n(T),\theta_m(T))^p : n,m \ge 1\}$$

is uniformly integrable, because its (q/p)th moments are uniformly bounded and $q/p > 1$. The almost-sure convergence established above and the Vitali convergence theorem therefore give

$$\mathop{\mathbb{E}}_{\xi} d_\infty^{\text{op}}(\theta_n(T),\theta_m(T))^p \longrightarrow 0$$

as $\min\{n,m\} \to \infty$, conditional on $(\Gamma,S) = (\gamma,s)$.

Conclusion. Couple μ_n and μ_m as follows: draw a single pair (Γ,S), and then, conditionally on it, draw the two initializations ξ_n and ξ_m independently. Since W_p is an infimum over couplings,

$$W_p(\mu_n,\mu_m;d_\infty^{\text{op}})^p \le \mathop{\mathbb{E}}_{\Gamma,S}\mathop{\mathbb{E}}_{\xi} d_\infty^{\text{op}}(\theta_n,\theta_m)^p.$$

The previous step shows that the inner expectation tends to 0 as $\min\{n, m\} \to \infty$, for every fixed (γ, s) in the support of (Γ, S). We now bound the inner expectation uniformly in (γ, s) and in n, m. Let $C < \infty$ denote the left-hand side of (46). Since $p < q$, Lyapunov's inequality — that is, Jensen's inequality applied to the concave map $t \mapsto t^{p/q}$ on $[0, \infty)$ — gives

$$\mathop{\mathbb{E}}_{\xi} d_\infty^{\mathrm{op}}(\theta_n, \theta_m)^p \le \left(\mathop{\mathbb{E}}_{\xi} d_\infty^{\mathrm{op}}(\theta_n, \theta_m)^q \right)^{p/q} \le C^{p/q},$$

a bound independent of (γ, s) and of n, m. So the dominated convergence theorem applies and yields

$$\mathop{\mathbb{E}}_{\Gamma, S} \mathop{\mathbb{E}}_{\xi} d_\infty^{\mathrm{op}}(\theta_n, \theta_m)^p \longrightarrow 0,$$

and hence $W_p(\mu_n, \mu_m; d_\infty^{\mathrm{op}}) \to 0$, whenever $\min\{n, m\} \to \infty$. Therefore, for every $\varepsilon > 0$, there exists $N \ge 1$ such that

$$W_p(\mu_n, \mu_m; d_\infty^{\mathrm{op}}) < \varepsilon$$

whenever $n, m \ge N$. If $n \ge N$, this bound holds for every $m \ge n$. Taking the supremum over $m \ge n$ proves (45). □

Corollary 18 (Size generalization on μP-trained 1-layer MLPs)**.** *Assume the hypotheses of Proposition 17 for some $q > 2$. Let the activation σ and all components of the duplication-compatible GMN f^Θ be Lipschitz. Let $X \subseteq \mathbb{R}^{d_{\mathrm{in}}}$ be compact, let the target functional $F^* \colon \mathcal{C}(X, \mathbb{R}^{d_{\mathrm{out}}}) \to \mathbb{R}^k$ be Lipschitz with respect to the uniform norm, and let the loss $\ell \colon \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ be Lipschitz in each argument. Define the risk*

$$\mathcal{R}_n(f^\Theta) := \mathop{\mathbb{E}}_{\theta \sim \mu_n} \ell\left(f^\Theta(\theta), F^*(\mathcal{R}(\theta))\right),$$

where $\mathcal{R}_n$ (subscripted, the risk at width n) and $\mathcal{R}$ (unsubscripted, the realization map of Proposition 2) denote different objects sharing the letter $\mathcal{R}$. Then

$$\sup_{m \ge n} \left| \mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta) \right| \longrightarrow 0 \qquad \text{as } n \to \infty.$$

Proof. The proof of Proposition 2(i) shows that the realization map $\mathcal{R}$ is L_R-Lipschitz on every ball of radius R with respect to $\|\cdot\|_{V_\infty, \mathrm{op}}$. For $L = 1$, the recursions defining C_ℓ and C'_ℓ in that proof show that C_0 and C'_1 are independent of R, that C_1 grows at most linearly in R, and hence that $L_R = \sqrt{d_{\mathrm{out}}}\, C'_2$ grows at most linearly in R. Applying this ballwise estimate with $R = \max\{\|\theta\|_{V_\infty, \mathrm{op}}, \left\|\widetilde{\theta}\right\|_{V_\infty, \mathrm{op}}\}$ shows that there is a constant $C_{\mathcal{R}} < \infty$ such that

$$\left\| \mathcal{R}(\theta) - \mathcal{R}(\widetilde{\theta}) \right\|_\infty \le C_{\mathcal{R}} \left(1 + \|\theta\|_{V_\infty, \mathrm{op}} + \left\|\widetilde{\theta}\right\|_{V_\infty, \mathrm{op}} \right) \left\| \theta - \widetilde{\theta} \right\|_{V_\infty, \mathrm{op}}$$

for every $\theta, \widetilde{\theta} \in V_\infty$. Both $\mathcal{R}$ and the normalized spectral norm are invariant under hidden-neuron permutations. Applying the preceding inequality to $g \cdot \theta$ and $\widetilde{\theta}$, and then taking the infimum over $g \in \mathsf{G}_\infty$, gives

$$\left\| \mathcal{R}(\theta) - \mathcal{R}(\widetilde{\theta}) \right\|_\infty \le C_{\mathcal{R}} \left(1 + \|\theta\|_{V_\infty, \mathrm{op}} + \left\|\widetilde{\theta}\right\|_{V_\infty, \mathrm{op}} \right) d_\infty^{\mathrm{op}}(\theta, \widetilde{\theta}).$$

By Proposition 2(ii), the duplication-compatible GMN is L_Θ-Lipschitz with respect to $\|\cdot\|_{V_\infty, 1}$. For 1-layer input MLPs, Proposition 10 gives $\|\theta'\|_{V_\infty, 1} \le \kappa \|\theta'\|_{V_\infty, \mathrm{op}}$ for every $\theta' \in V_\infty$, where $\kappa := \max\{1, \sqrt{d_{\mathrm{in}}}, \sqrt{d_{\mathrm{out}}}\}$. The invariance of the GMN under hidden-neuron permutations then gives

$$\left\| f^\Theta(\theta) - f^\Theta(\widetilde{\theta}) \right\|_2 \le \kappa L_\Theta d_\infty^{\mathrm{op}}(\theta, \widetilde{\theta}).$$

Write L_{loss} for a Lipschitz constant of ℓ in each argument and L_* for one of F^*, and set $h(\theta) := \ell\left(f^\Theta(\theta), F^*(\mathcal{R}(\theta))\right)$. The preceding bounds and the triangle inequality give a constant $C < \infty$ such that

$$\left| h(\theta) - h(\widetilde{\theta}) \right| \le C \left(1 + \|\theta\|_{V_\infty, \mathrm{op}} + \left\|\widetilde{\theta}\right\|_{V_\infty, \mathrm{op}} \right) d_\infty^{\mathrm{op}}(\theta, \widetilde{\theta}).$$

Taking $\widetilde{\theta} = 0$ and using the assumed qth-moment bound shows that h is integrable under every μ_n.

Set $p_* := q/(q-1)$. Because $q > 2$, we have $1 < p_* < 2 < q$, so Proposition 17 applies with $p = p_*$. Since we assumed $\sup_{\gamma,s} \sup_{n\geq 1} \mathbb{E}_{\xi_n} \|\theta_n\|^q_{V_n,\mathrm{F}} < \infty$. Combined with the domination $\|\theta\|_{V_\infty,\mathrm{op}} \leq \sqrt{2(L+1)}\,\|\theta\|_{V_\infty,\mathrm{F}}$ of Proposition 9, it gives

$$\sup_{n\geq 1} \mathop{\mathbb{E}}_{\theta\sim\mu_n} \|\theta\|^q_{V_\infty,\mathrm{op}} < \infty.$$

For every coupling π of μ_n and μ_m, Hölder's inequality with conjugate exponents q and p_* therefore gives a constant $C' < \infty$ independent of n, m and π such that

$$\left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right| \leq \mathop{\mathbb{E}}_{\pi} \left|h(\theta) - h(\widetilde{\theta})\right| \leq C' \left(\mathop{\mathbb{E}}_{\pi} d^{\mathrm{op}}_\infty(\theta,\widetilde{\theta})^{p_*}\right)^{1/p_*}.$$

Taking the infimum over π gives

$$\left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right| \leq C' W_{p_*}(\mu_n,\mu_m; d^{\mathrm{op}}_\infty).$$

The uniform convergence in (45) now proves the result. □

E.2.3 Obstruction on MLP with larger depths

The proof in Appendix E.2.2 crucially relies on the fact that the MLP is 1-layer, and in particular that there is no hidden-to-hidden weight matrix. We now show the obstruction to generalize the result to MLPs with ≥ 3 layers. We can actually prove that μP-trained MLPs with ≥ 3 layers, $W_1(\mu_n,\mu_{2n}; d^{\mathrm{op}}_\infty) \not\to 0$ at initialization.

For simplicity, consider randomly initialized n-width and $2n$-width MLPs with ≥ 3 layers, and let μ_n and μ_{2n} denote the corresponding weight distributions. Focus on one hidden-to-hidden layer, whose width-n and width-$2n$ matrices are respectively $n\times n$ and $2n\times 2n$. We first map the width-n weights to get an equivalent width-$2n$ model: $W_n \mapsto W_n \otimes \frac{1}{2}\mathbf{1}_2\mathbf{1}_2^\top$. The resultant matrix has rank at most n, so its $(n+1)$-th singular value is zero.

The width-$2n$ matrix W_{2n} has i.i.d. entries with variance $1/(2n)$ at initialization. Standard random matrix theory results show that the empirical distribution of its squared singular values converges to the Marchenko–Pastur distribution supported on $[0,4]$. Choose $c > 0$ small enough so that this limiting distribution assigns mass greater than $1/2$ to (c^2,∞). Then, with probability tending to one, more than n singular values of the $2n\times 2n$ matrix exceed c. Equivalently, the $(n+1)$-st singular value is bounded below by c with probability tending to one. Finally, Weyl's inequality gives, for every hidden-unit permutation g at width $2n$,

$$\left\|W_{2n} - g\cdot\left(W_n\otimes\tfrac{1}{2}\mathbf{1}_2\mathbf{1}_2^\top\right)\right\| \geq s_{n+1}\left(W_{2n}\right) - 0 \geq c > 0$$

on this high-probability event. This lower bound is independent of the coupling of the two initialized networks, because it depends only on the width-$2n$ marginal. Hence, for every coupling π of μ_n and μ_{2n},

$$\mathop{\mathbb{E}}_{\pi} d^{\mathrm{op}}_\infty(\theta_n,\theta_{2n}) \geq c\,\mathbb{P}\left(s_{n+1}(W_{2n}) \geq c\right).$$

Taking the infimum over couplings gives

$$\liminf_{n\to\infty} W_1(\mu_n,\mu_{2n}; d^{\mathrm{op}}_\infty) \geq c > 0,$$

so the convergence in (45) fails at initialization.

For trained weights at fixed positive time, it is unknown whether the same obstruction persists. On the one hand, empirically it is observed that sometimes, the trained weights have a low-rank structure, which may remove the obstruction; On the other hand, the existing μP theory does not seem to give any guarantee of such closeness.

E.3 Proof of Theorem 5

The previous subsection compared independently trained weights through the normalized spectral distance d^{op}_∞. That strategy works for 1-layer MLPs under μP, where trained weights of different

widths are provably close in d_∞^{op}, but it breaks down as soon as the input MLP has a hidden-to-hidden layer: by Appendix E.2.3, weights of different widths stay separated in d_∞^{op} already at initialization.

This does not mean that size generalization itself fails at depth ≥ 2; it only means that this particular route to it is unavailable. The reason is that d_∞^{op} is not a faithful measure of the distance that the GMN class prescribes. We proved that the matrix-product GMNs (or duplication-compatible GMNs in the 1-layer MLP case) are Lipschitz with respect to d_∞^{op}, so that closeness in d_∞^{op} implies closeness of the GMN outputs. But the converse is not true: weights can produce close GMN outputs while remaining far apart in d_∞^{op}. Consequently $W_1(\mu_n, \mu_m; d_\infty^{\mathrm{op}}) \to 0$ is a sufficient but not a necessary condition for the size-generalization gap $\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)$ to vanish.

This suggests measuring the two weight distributions in a distance generated by the GMN class itself. Concretely, let $\mathcal{A} \subset \mathbb{R}^q$ be a compact parameter set for the matrix-product GMN, and for $\Theta \in \mathcal{A}$ write f_n^Θ for its width-n instantiation. We consider the following distance between the trained weight distributions μ_n and μ_m:

$$d_{\mathcal{A}}(\mu_n, \mu_m) := \sup_{\Theta \in \mathcal{A}} \left| \mathop{\mathbb{E}}_{\theta \sim \mu_n} \ell(f_n^\Theta(\theta), f_n^*(\theta)) - \mathop{\mathbb{E}}_{\theta' \sim \mu_m} \ell(f_m^\Theta(\theta'), f_m^*(\theta')) \right|.$$

This is the *integral probability metric* (IPM) generated by the width-indexed test class of composed maps $\theta \mapsto \ell(f_n^\Theta(\theta), f_n^*(\theta))$, $\Theta \in \mathcal{A}$. IPMs are a standard way of comparing probability measures, taking the supremum of $|\mathbb{E}_\mu h - \mathbb{E}_\nu h|$ over a class of test functions h. Total variation, the Wasserstein-1 distance, and maximum mean discrepancy are examples of IPM with different choices of the test class (Müller, 1997). It defines a pseudometric on the measures for which the integrals exist.

By construction $d_{\mathcal{A}}$ dominates the size-generalization gap, since $\left|\mathcal{R}_n(f^\Theta) - \mathcal{R}_m(f^\Theta)\right| \leq d_{\mathcal{A}}(\mu_n, \mu_m)$ for every fixed $\Theta \in \mathcal{A}$. The goal of this subsection is therefore to show that $d_{\mathcal{A}}(\mu_n, \mu_m) \to 0$ as $n, m \to \infty$.

High-level idea of the proof. In Appendix E.1, we reviewed that the training of MLP under μP of fixed number of training steps can be written with a tensor program, with the infinite-width limit precisely described. The key observation we use here is that, evaluating the matrix-product GMN on those weights does not leave this class. For a fixed parameter Θ, the only way f^Θ touches a width-n weight matrix is by multiplying it, or its transpose, against a vector assembled from node features with fixed-dimensional coefficient matrices. Every such operation is a legal tensor-program instruction. Appending these instructions, message-passing round by message-passing round, to the program that produced θ_n therefore realizes the computation $f_n^\Theta(\theta_n)$ inside a single tensor program. Assuming that the target summary f_n^* and the loss ℓ are likewise expressible as tensor-program operations, the scalar $\ell(f_n^\Theta(\theta_n), f_n^*(\theta_n))$ is an output of one finite tensor program. The tensor program machinery then gives a deterministic limit for this scalar as $n \to \infty$.

The remaining steps are technical. We upgrade this convergence to that of the expected risk $\mathbb{E}_{\theta_n \sim \mu_n}[\ell(f_n^\Theta(\theta_n), f_n^*(\theta_n))]$, and furthermore, to uniform convergence over $\Theta \in \mathcal{A}$. We now carry out this proof technique in detail.

Evaluation of matrix-product GMN on trained weights as a tensor program. We first spell out why evaluating a matrix-product GMN (with fixed parameter Θ) on the trained weights can be appended to the same tensor program that produced those weights.

Following the original tensor program convention, we assume the MLP has homogeneous hidden-layer widths ($n_1 = \cdots = n_L = n$). This is purely for simplified notation, and extending to heterogeneous widths is straightforward. Throughout, the GMN parameter Θ is fixed, so its entries can be hardcoded into the OUTERNONLIN instructions, and no separate constant inputs are needed.

We append the GMN evaluation by induction over message-passing rounds.

Initialization of node features at $t = 0$. Each hidden-layer node feature coordinate $H_{:,a}^{(\ell),[0]} \in \mathbb{R}^n$ for $1 \leq \ell \leq L$, $1 \leq a \leq d_h$ is produced as a program vector by a single OUTERNONLIN instruction with $r = 0$, applying entrywise to the trained bias $b^{(\ell)}(T) \in \mathbb{R}^n$, which is an existing program vector. Indeed, for the initialization $h_i^{(\ell),[0]} = \psi_v\left(b_i^{(\ell)}\right)$ of the simple GMN, the instruction is the

one whose nonlinearity $\boldsymbol{x}_j \mapsto (\psi_v)_a \left(\left(b^{(\ell)}(T) \right)_j \right)$ reads off the coordinate of the row $\boldsymbol{x}_j$ holding $b^{(\ell)}(T)$, applies the ath output coordinate of ψ_v to it, and ignores all other arguments. Since ψ_v is an MLP, it is Lipschitz, hence pseudo-Lipschitz, and the extended program still satisfies Assumption 1.

The input node features $H^{(0),[0]} \in \mathbb{R}^{d_{\text{in}} \times d_h}$ form a constant matrix. The output node features $H^{(L+1),[0]} \in \mathbb{R}^{d_{\text{out}} \times d_h}$ are fixed-dimensional; their entries are produced as program scalars from the existing program scalars $b_i^{(L+1)}(T)$, $i \in [d_{\text{out}}]$, using OUTERNONLIN and AVG.

Message-passing round $t \geq 0$. Assume that the columns of $H^{(\ell),[t]}$ have already been produced as program vectors for every hidden layer $\ell \in [1, L]$, and the entries of $H^{(0),[t]}$ and $H^{(L+1),[t]}$ have been produced as program scalars.

We first produce, for $\ell \in [1, L+1], 1 \leq a \leq d_h$,

$$U_{:,a}^{(\ell),\text{in},[t]} = \sum_{b=1}^{d_h} H_{:,b}^{(\ell-1),[t]} \, (A_\ell^{\text{in}})_{ba},$$

which is a program vector in $\mathbb{R}^n$ for $\ell \neq 1$, and a set of program scalars for $\ell = 1$. For $\ell \geq 2$, this is produced by a OUTERNONLIN instruction on existing program vectors $H_{:,b}^{(\ell-1),[t]}$, with the nonlinearity $\boldsymbol{x}_j \mapsto \sum_{b=1}^{d_h} (\boldsymbol{x}_j)_b \, (A_\ell^{\text{in}})_{ba}$. The scalar case $\ell = 1$ is the same finite linear combination at the level of program scalars. The entries $(A_\ell^{\text{in}})_{ba}$ are fixed numbers that can be hardcoded into the OUTERNONLIN instruction.

We then produce the aggregate of incoming messages at each node. When $2 \leq \ell \leq L$, we produce a program vector

$$Z_{:,a}^{(\ell),\text{in},[t]} = W^{(\ell)}(T) U_{:,a}^{(\ell),\text{in},[t]} \in \mathbb{R}^n,$$

for $a \in [1, d_h]$, by applying MATMUL to the initial i.i.d. matrix part and using OUTERNONLIN and AVG for the finite outer-product terms representing the gradient updates during training. When $\ell = 1$, produce

$$Z_{:,a}^{(1),\text{in},[t]} = \sum_{j=1}^{d_{\text{in}}} W_{:,j}^{(1)}(T) \, U_{j,a}^{(1),\text{in},[t]},$$

for $a \in [1, d_h]$, which is a OUTERNONLIN instruction applied to existing program vectors $W_{:,j}^{(1)}(T)$ and program scalars $U_{j,a}^{(1),\text{in},[t]}$. When $\ell = L+1$, first produce a program vector

$$Y_{i,:,a}^{(L+1),\text{in},[t]} = \overline{W}_{i,:}^{(L+1)}(T) \odot U_{:,a}^{(L+1),\text{in},[t]},$$

for each $i \in [d_{\text{out}}]$ and $a \in [d_h]$, by OUTERNONLIN with $r = 0$, using that $\overline{W}_{i,:}^{(L+1)}(T)$ and $U_{:,a}^{(L+1),\text{in},[t]}$ are existing program vectors. We highlight here that the μP scaling ensures that the rescaled weight $\overline{W}_{i,:}^{(L+1)}(T)$ has order-one entries throughout training and hence can be appended to the tensor program as a program vector. Then, apply AVG to obtain

$$Z_{i,a}^{(L+1),\text{in},[t]} = \frac{1}{n} \sum_{\alpha=1}^{n} \overline{W}_{i\alpha}^{(L+1)}(T) U_{\alpha,a}^{(L+1),\text{in},[t]} = \sum_{\alpha=1}^{n} W_{i\alpha}^{(L+1)}(T) U_{\alpha,a}^{(L+1),\text{in},[t]}.$$

The outgoing aggregation to form $Z^{(\ell),\text{out},[t]}$ can be performed in the same way, so we leave it out for brevity.

Finally, the node update is row-wise. For each $\ell \in [1, L], 1 \leq a \leq d_h$, produce

$$H_{:,a}^{(\ell),[t+1]} = \psi_{h,a} \left(H^{(\ell),[t]}, Z^{(\ell),\text{in},[t]}, Z^{(\ell),\text{out},[t]} \right),$$

by OUTERNONLIN with $r = 0$ applied to existing program vectors $H_{:,b}^{(\ell),[t]}$, $Z_{:,b}^{(\ell),\text{in},[t]}$, and $Z_{:,b}^{(\ell),\text{out},[t]}$ for $b \in [d_h]$. ψ_h is an MLP, hence pseudo-Lipschitz, so the extended program still satisfies Assumption 1. The node update for $\ell = 0$ and $\ell = L+1$ is similarly produced as program scalars by OUTERNONLIN and AVG.

This completes the induction for one message-passing round, and repeating it for the fixed number of rounds gives all final node features.

Graph-level readout. For each $\ell \in [1, L]$ and $1 \le a \le d_h$, produce a program vector

$$R^{(\ell)}_{:,a} = \psi_{r,a}\left(H^{(\ell),[T]}\right) \in \mathbb{R}^n,$$

by OUTERNONLIN with $r = 0$, and then apply AVG to obtain

$$r^{(\ell)}_a = \frac{1}{n}\sum_{\alpha=1}^{n} R^{(\ell)}_{\alpha,a}.$$

The case $\ell = 0$ and $\ell = L + 1$ is similarly produced as program scalars by OUTERNONLIN and AVG.

The final readout is produced by applying ρ to the program scalars $r^{(\ell)}_a$ for $\ell \in [0, L + 1]$ and $a \in [d_h]$. ρ is an MLP and hence again pseudo-Lipschitz, so the extended program still satisfies Assumption 1.

Evaluation of the target and loss as a tensor program. Next, we want to further append the evaluation of the loss $\ell(f^{\Theta}_n(\theta_n), f^*_n(\theta_n))$ to the tensor program. With the above, this will ensure the full computation "train the input network, evaluate the matrix-product GMN, evaluate the target, and evaluate the loss" is altogether a finite tensor program.

This requires the following assumption on the target functions $(f^*_n)_n$ and the loss ℓ.

Assumption 3. (i) For each width n, the target function $f^*_n : V_n \to \mathbb{R}^k$ is deterministic and its computation can be expressed with finitely many instructions of AVG, OUTERNONLIN, and MATMUL, on the following inputs:

- Matrix inputs: the trained hidden-to-hidden matrices $W^{(\ell)}(T) \in \mathbb{R}^{n\times n}$ for $2 \le \ell \le L$.
- Vector inputs: the columns of input weights $W^{(1)}(T) \in \mathbb{R}^{n\times d_{\text{in}}}$, and the rows of scaled output weights $\overline{W}^{(L+1)}(T) \in \mathbb{R}^{d_{\text{out}}\times n}$, and the trained hidden biases $b^{(\ell)}(T) \in \mathbb{R}^n$ for $1 \le \ell \le L$.
- Scalar inputs: the trained output bias $b^{(L+1)}(T) \in \mathbb{R}^{d_{\text{out}}}$, and any other fixed, width-independent scalars.

Every nonlinearity used in a OUTERNONLIN instruction in this computation is assumed to be pseudo-Lipschitz in the sense of Assumption 1(d), and fixed independently of the width.

(ii) The loss $\ell\colon \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ is Lipschitz.

(iii) For some $q > 1$, the loss scalars have a width-uniform L^q bound,

$$\sup_{\Theta\in\mathcal{A}} \sup_{(\gamma,s)\in\operatorname{supp}(\Gamma,S)} \sup_{n\ge 1} \mathbb{E}_{\xi_n} \left|\ell\left(f^{\Theta}_n(\theta_n), f^*_n(\theta_n)\right)\right|^q < \infty.$$

The assumption on the target functions $(f^*_n)_n$ in Assumption 3(i) seems convoluted. We discuss when and when not it is satisfied.

Remark 1 (Satisfaction of the target function assumption). A convenient sufficient condition is *exact uniform representability* of the target family $(f^*_n)_n$ by the matrix-product GMN. In that case, the evaluation of f^*_n can be replaced by the evaluation of the matrix-product GMN that represents it at width n, which we proved above is a tensor program. With some additional work, the result might extend to target families that can be *uniformly approximated* by the matrix-product GMN across widths. The expressivity of the matrix-product GMN on MLPs of different widths is not yet well understood, and remains an important open question.

The assumption is also satisfied by targets obtained from finitely many evaluations of the input MLP itself. For example, fix width-independent inputs $x_1, \dots, x_m \in \mathbb{R}^{d_{\text{in}}}$ and let

$$f^*_n(\theta) = \Phi\left(\mathcal{R}(\theta)(x_1), \dots, \mathcal{R}(\theta)(x_m)\right),$$

where $\Phi\colon (\mathbb{R}^{d_{\text{out}}})^m \to \mathbb{R}^k$ is a fixed Lipschitz function. This covers, for instance, finite-validation-set generalization prediction for a Lipschitz objective.

By contrast, the assumption does not cover an arbitrary functional $F^*\colon \mathcal{C}(X, \mathbb{R}^{d_{\text{out}}}) \to \mathbb{R}^k$ applied to the realized function $\mathcal{R}(\theta)$. Such a functional may depend on the values of $\mathcal{R}(\theta)$ on infinitely many inputs, or on features that are not reducible to finitely many evaluations and width-independent nonlinearities. Although the weights determine the function $\mathcal{R}(\theta)$, this does not by itself give a finite tensor-program computation of $F^*(\mathcal{R}(\theta))$ from the weights. This is the main difference from the other size generalization result in Corollary 18, which only assumes that the function-space functional is Lipschitz with respect to the uniform norm.

Finally, we give the proof of the main result.

We first give a technical lemma that upgrades pointwise convergence to uniform convergence on compact sets.

Lemma 19 (Equicontinuous pointwise convergence on compact sets)**.** *Let (K, d) be a compact metric space, and let $g_n\colon K \to \mathbb{R}$ be an equicontinuous sequence of functions. If $g_n(x) \to g(x)$ for every $x \in K$, then g is continuous and $g_n \to g$ uniformly on K.*

Proof. Let $\varepsilon > 0$. By equicontinuity, choose $\delta > 0$ such that $d(x, y) < \delta$ implies

$$|g_n(x) - g_n(y)| < \varepsilon/3 \qquad \text{for all } n.$$

Taking $n \to \infty$ in this inequality gives

$$|g(x) - g(y)| \le \varepsilon/3 \qquad \text{whenever } d(x, y) < \delta.$$

This proves continuity of g. Since K is compact, choose a finite δ-net $\{x_1, \dots, x_R\}$ of K. For each r, pointwise convergence gives N_r such that

$$|g_n(x_r) - g(x_r)| < \varepsilon/3 \qquad \text{for all } n \ge N_r.$$

Let $N := \max_{1 \le r \le R} N_r$. For any $n \ge N$ and any $x \in K$, choose r with $d(x, x_r) < \delta$. The triangle inequality gives

$$|g_n(x) - g(x)| \le |g_n(x) - g_n(x_r)| + |g_n(x_r) - g(x_r)| + |g(x_r) - g(x)| < \varepsilon.$$

Taking the supremum over $x \in K$ proves uniform convergence. □

Proposition 20. *Assume Assumption 3, and that the matrix-product GMN satisfies the hypotheses of Proposition 12. Assume also that the trained weights have a width-uniform moment of order $T + 1$ in the normalized spectral norm, T being the number of message-passing rounds:*

$$\sup_{n \ge 1} \mathop{\mathbb{E}}_{\Gamma, S} \mathop{\mathbb{E}}_{\xi_n} \|\theta_n\|_{V_n, \text{op}}^{T+1} < \infty. \tag{47}$$

Then

$$\sup_{m \ge n} d_{\mathcal{A}}(\mu_n, \mu_m) \to 0 \qquad \text{as } n \to \infty.$$

Proof. We prove convergence through a common infinite-width limit.

Almost sure convergence of loss scalars. We first fix (γ, s) in the support of (Γ, S) and condition on $(\Gamma, S) = (\gamma, s)$. This fixes the hyperparameters γ and the auxiliary training randomness s, so that the initialization is the only remaining source of randomness. Throughout the conditional argument, θ_n denotes $\mathsf{Train}_n(\gamma, s, \xi_n)$.

Fix $\Theta \in \mathcal{A}$. We have shown that the evaluation of the matrix-product GMN on the trained weights can be appended to the tensor program that produced those weights; furthermore, under Assumption 3, the evaluation of the target function on the trained weights, and the computation of the loss can all be appended to the same tensor program. Consequently, we have a finite tensor program where $\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right)$ is a scalar produced by it. The master theorem Theorem 14 therefore gives a deterministic conditional limit $L_\infty(\Theta; \gamma, s)$ such that

$$\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right) \xrightarrow[n \to \infty]{\text{a.s.}} L_\infty(\Theta; \gamma, s).$$

Convergence of expected risks. By assumption Assumption 3(iii), the qth absolute moment satisfies

$$\mathop{\mathbb{E}}_{\xi_n} \left|\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right)\right|^q < C$$

for some constant C independent of (γ, s) and uniformly in n. Since $q > 1$, the de la Vallée–Poussin criterion shows that the family $\{\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right) : n \geq 1\}$ is uniformly integrable with respect to ξ_n. The almost-sure convergence above and the Vitali convergence theorem therefore give

$$\mathop{\mathbb{E}}_{\xi_n} \ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right) \longrightarrow L_\infty(\Theta; \gamma, s).$$

We next pass from the conditional expectation to the expectation over (Γ, S). Jensen's inequality applied to the concave map $t \mapsto t^{1/q}$, gives

$$\left|\mathop{\mathbb{E}}_{\xi_n} \ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right)\right| \leq \mathop{\mathbb{E}}_{\xi_n} \left|\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right)\right| \leq \left(\mathop{\mathbb{E}}_{\xi_n} \left|\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right)\right|^q\right)^{1/q} \leq C^{1/q},$$

a bound independent of (γ, s) and n. The bounded convergence theorem over (Γ, S) yields

$$a_n(\Theta) := \mathop{\mathbb{E}}_{\Gamma,S} \mathop{\mathbb{E}}_{\xi_n} \ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right) \longrightarrow a_\infty(\Theta) := \mathop{\mathbb{E}}_{\Gamma,S} L_\infty(\Theta; \Gamma, S).$$

Uniformity over Θ. We first prove that the family $\{a_n : n \geq 1\}$ is Lipschitz on $\mathcal{A}$ with a common Lipschitz constant.

By Proposition 12 and the Lipschitzness of ℓ, for all $\Theta, \widetilde{\Theta} \in \mathcal{A}$,

$$\left|\ell\left(f_n^{\Theta}(\theta_n), f_n^*(\theta_n)\right) - \ell\left(f_n^{\widetilde{\Theta}}(\theta_n), f_n^*(\theta_n)\right)\right| \leq M_n \left\|\Theta - \widetilde{\Theta}\right\|_2,$$

where $M_n := \Lambda c_{\mathcal{A},T} \left(1 + \|\theta_n\|_{V_n,\mathrm{op}}\right)^{T+1}$, Λ is the Lipschitz constant for the loss in Assumption 3(ii), and $c_{\mathcal{A},T}$ is the constant from Proposition 12.

Since $(1+x)^{T+1} \leq 2^{T+1}(1 + x^{T+1})$ for $x \geq 0$, the assumed moment bound (47) gives

$$\sup_{n\geq 1} \mathop{\mathbb{E}}_{\Gamma,S} \mathop{\mathbb{E}}_{\xi_n} M_n \leq 2^{T+1} \Lambda c_{\mathcal{A},T} \left(1 + \sup_{n\geq 1} \mathop{\mathbb{E}}_{\Gamma,S} \mathop{\mathbb{E}}_{\xi_n} \|\theta_n\|_{V_n,\mathrm{op}}^{T+1}\right) := C_{\mathcal{A}} < \infty.$$

Taking expectations in the bound above gives, for all $\Theta, \widetilde{\Theta} \in \mathcal{A}$,

$$\left|a_n(\Theta) - a_n(\widetilde{\Theta})\right| \leq C_{\mathcal{A}} \left\|\Theta - \widetilde{\Theta}\right\|_2 \qquad \text{for all } n.$$

By the pointwise convergence already proved, the same Lipschitz bound holds for a_∞ after taking $n \to \infty$.

Since $\mathcal{A}$ is compact, the common Lipschitz bound and Lemma 19 upgrade the pointwise convergence $a_n(\Theta) \to a_\infty(\Theta)$ to

$$\sup_{\Theta\in\mathcal{A}} |a_n(\Theta) - a_\infty(\Theta)| \longrightarrow 0.$$

Finally,

$$\begin{aligned}\sup_{m\geq n} d_{\mathcal{A}}(\mu_n, \mu_m) &= \sup_{m\geq n} \sup_{\Theta\in\mathcal{A}} |a_n(\Theta) - a_m(\Theta)| \\ &\leq \sup_{\Theta\in\mathcal{A}} |a_n(\Theta) - a_\infty(\Theta)| + \sup_{m\geq n} \sup_{\Theta\in\mathcal{A}} |a_m(\Theta) - a_\infty(\Theta)| \longrightarrow 0.\end{aligned}$$

□

Corollary 21 (Size generalization for matrix-product GMNs under μP)**.** *Assume Assumption 3 and the remaining hypotheses of Proposition 20. For each fixed $\Theta \in \mathcal{A}$, define*

$$\mathcal{R}_n(f^{\Theta}) := \mathop{\mathbb{E}}_{\theta\sim\mu_n} \ell\left(f_n^{\Theta}(\theta), f_n^*(\theta)\right).$$

Then

$$\sup_{m\geq n} \left|\mathcal{R}_n(f^{\Theta}) - \mathcal{R}_m(f^{\Theta})\right| \longrightarrow 0 \qquad \textit{as } n \to \infty.$$

Proof. For every $m \geq n$ and every fixed $\Theta \in \mathcal{A}$,

$$\left|\mathcal{R}_n(f^{\Theta}) - \mathcal{R}_m(f^{\Theta})\right| \leq d_{\mathcal{A}}(\mu_n, \mu_m).$$

Taking the supremum over $m \geq n$ and applying Proposition 20 proves the claim. □