PERMUTATION-EQUIVARIANT FLOW MATCHING FOR ALIGNMENT-FREE NEURAL WEIGHT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A trained neural network can be represented by a parameter vector in high dimensions. Learning distributions over these vectors enables the generation of new models across various tasks and architectures. A central challenge is permutation symmetry: permuting hidden neurons can produce distant parameter vectors representing the same function. This introduces variations that a generative model must account for when learning from trained networks. Existing methods typically address this using networks derived from a common base model or costly approximate neuron alignment. We instead parameterize a flow-matching velocity field with a permutation-equivariant Graph Meta Network, enabling direct learning from independently trained networks without alignment. Extensive experiments show that our method closely reproduces the joint statistics of accuracy, functional similarity, and weight similarity of independently trained collections, providing evidence of generation beyond checkpoint memorization. A single conditional model also generates task-specific networks on heterogeneous architectures and generalizes to unseen hidden-width configurations. On a tabular domain-shift task, intermediate conditioning produces individual networks with performance comparable to logit ensembles across both domains. Taken together, our results show how permutation equivariance enables learning from diverse collections of independently trained networks without permutation alignment.

1 INTRODUCTION

Trained network weights have emerged as a new data modality (Unterthiner et al., 2020; Eilertsen et al., 2020; Schürholt et al., 2021; Navon et al., 2023; 2024; Lim et al., 2024; Putterman et al., 2024; Gelberg et al., 2025; Dayan et al., 2026; Knyazev et al., 2025; Tran et al., 2025; Kofinas et al., 2024; Kalogeropoulos et al., 2024). Weight-space generative modeling promises a paradigm shift: rather than optimize neural networks, we could sample them directly from a learned distribution. Weight generation, rather than optimization, may unlock powerful capabilities, most notably conditional generation, allowing us to synthesize networks tailored to specific datasets, architectures, or domains (Wang et al., 2026; Han et al., 2026). Despite this potential, learning the distribution of independently trained neural networks remains a challenge.

Ideally, we wish to train a generative model on large-scale network collections, spanning diverse tasks and architectures, to capture the distribution of diverse solutions. However, two functionally identical networks can have different weight matrices due to arbitrary hidden-neuron ordering (Figure 1). This is a practical concern as repositories like Hugging Face (Wolf et al., 2020) host diverse trained networks without a common ordering or shared initialization. As such, weight-space symmetries, transformations that change weights yet preserve the function, especially permutation symmetries, are ubiquitous and thus form a fundamental obstacle in weight-space generation.

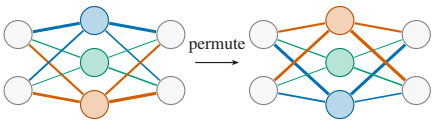

Figure 1: Hidden-unit permutations preserve the network function. Colors track each hidden unit and its incident weights.

To avoid this issue, prior works alter the network collections used for training. Post-hoc approaches approximately align compatible training networks to a common reference (Ainsworth et al., 2022). A priori methods instead create artificially the training collection so that alignment is not needed,

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

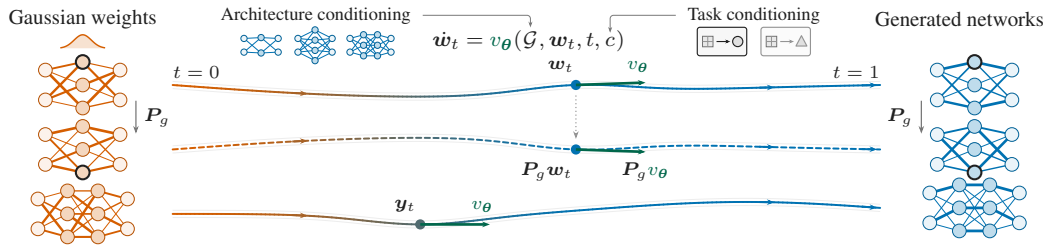

Figure 2: **Method overview.** A shared velocity field $v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}_t, t, c)$, trained on unaligned networks, generates weights conditioned on architecture $\mathcal{G}$ and task c. States $\boldsymbol{w}_t$ and $\boldsymbol{y}_t$ belong to different architectures. A hidden-unit permutation $\boldsymbol{P}_g$ permutes the trajectory and its velocities, producing functionally equivalent networks (upper pair).

typically by sampling checkpoints along the optimization trajectory of a single network (Wang et al., 2024; 2025; Erkoç et al., 2023). These strategies require either compatible architectures for alignment or control over how the training collection is constructed, restricting their applicability to existing repositories of independently trained networks. Many generators impose further limitations through fixed-dimensional parameterizations, restricting their direct reuse across architectures (Wang et al., 2024; 2025; Erkoç et al., 2023; Schürholt et al., 2022; 2024; Gupta et al., 2026). Moreover, sample efficiency and memorization remain challenges even within these restricted settings (Zeng et al., 2026; Navon et al., 2023; 2024; Shamsian et al., 2024).

We advocate a switch to equivariant tools, which are natural for symmetries. Following recent advances in graph metanetworks (Lim et al., 2024; Knyazev et al., 2025; Kofinas et al., 2024; Kalogeropoulos et al., 2024), we represent neural networks as graphs, adopt Flow Matching (FM) (Lipman et al., 2023), and parameterize the velocity field with a Graph Meta Network (GMN) designed to be equivariant with weight-space permutations (Figure 2). We motivate this inductive bias by showing that permutation-invariant initialization and equivariant training yield a permutation-equivariant optimal FM velocity field. On up to 100K independently trained networks, our model generates networks with joint accuracy and similarity statistics much closer to the training network distribution than existing works. We further show that our method can interpolate across domain shifts, process heterogeneous architectures, and generate weights at unseen hidden widths.

Our main contributions can be summarized as follows:

1. **Alignment-free generation beyond memorization.** We parameterize a flow-matching velocity field with a permutation-equivariant GMN to learn directly from independently trained networks without alignment. Our proposed joint quality measure shows that generated networks closely match independently trained networks in performance and similarity, providing evidence against checkpoint memorization.
2. **Heterogeneous architecture generation.** A single flow generates weights across heterogeneous architectures and generalizes to unseen widths, without retraining or architecture-specific changes to the generator.
3. **Conditional weight-space traversal.** We demonstrate conditional generation across diverse tasks and domain shifts, showing that our flow generalizes to unseen conditions and synthesizes single unified models on par with logit ensembling.

2 Preliminaries

Weight space permutation symmetries. Consider a multilayer perceptron (MLP) $F(\boldsymbol{x}; \boldsymbol{w})$ with L hidden layers with widths $d_0, \ldots, d_{L+1}$, where $\boldsymbol{x} \in \mathbb{R}^{d_0}$ is the input and $\boldsymbol{w}$ is the vectorized network's weights and biases. The weight and bias of layer ℓ are given by $\boldsymbol{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $\boldsymbol{b}_\ell \in \mathbb{R}^{d_\ell}$. For each hidden layer, let $\boldsymbol{P}_\ell$ be a permutation matrix, and fix $\boldsymbol{P}_0 = \boldsymbol{I}_{d_0}$ and $\boldsymbol{P}_{L+1} = \boldsymbol{I}_{d_{L+1}}$ to identity matrices to preserve the input and output coordinates. Changing parameters to

$$\boldsymbol{W}'_\ell = \boldsymbol{P}_\ell \boldsymbol{W}_\ell \boldsymbol{P}_{\ell-1}^\top \quad \text{and} \quad \boldsymbol{b}'_\ell = \boldsymbol{P}_\ell \boldsymbol{b}_\ell, \qquad \text{for } \ell = 1, \ldots, L+1, \tag{1}$$

reorders hidden neurons and their incoming and outgoing weights. As elementwise activations commute with permutations, distinct parameters $\boldsymbol{w}'$ can represent the same function (Hecht-Nielsen, 1990; Navon et al., 2023)

$$F(\boldsymbol{x}; \boldsymbol{w}') = F(\boldsymbol{x}; \boldsymbol{w}), \qquad \forall \boldsymbol{x} \in \mathbb{R}^{d_0}. \tag{2}$$

The same goes for convolutional networks and Transformers (Kofinas et al., 2024; Lim et al., 2024).

Graph metanetworks. Graph metanetworks view a neural network as a *parameter graph* $\mathcal{G}$: nodes correspond to neurons or channels, and edges carry the associated weights (Lim et al., 2024; Kofinas et al., 2024; Kalogeropoulos et al., 2024; Bar-Shalom et al., 2026). Added positional encodings identify structural roles, e.g., layer membership. A Graph Neural Network (GNN) whose message-passing layers naturally respect weight-space symmetries processes this graph and outputs either edge features (e.g., new weights) or a global output (e.g., the network's accuracy on a dataset). The resulting metanetwork is equivariant to weight-space permutations, and its shared graph operations allow one model to process heterogeneous architectures.

Flow Matching. Flow Matching (FM) (Lipman et al., 2023) learns a velocity field $v_{\boldsymbol{\theta}}$ via a neural network with parameters $\boldsymbol{\theta}$, to transport a source distribution p_0 to target p_1. With independent coupling and linear interpolation: drawing $\boldsymbol{w}_0 \sim p_0$, $\boldsymbol{w}_1 \sim p_1$, and $t \sim \mathcal{U}(0, 1)$ independently, we define $\boldsymbol{w}_t = (1 - t)\boldsymbol{w}_0 + t\boldsymbol{w}_1$. The network is trained by minimizing the flow matching objective

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{t, \boldsymbol{w}_0, \boldsymbol{w}_1} \left[\| v_{\boldsymbol{\theta}}(\boldsymbol{w}_t, t) - (\boldsymbol{w}_1 - \boldsymbol{w}_0) \|_2^2 \right]. \tag{3}$$

At inference, we draw $\boldsymbol{w}_0 \sim p_0$ and integrate the following learned velocity field from $t = 0$ to 1

$$\dot{\boldsymbol{w}}_t = \frac{d\boldsymbol{w}_t}{dt} = v_{\boldsymbol{\theta}}(\boldsymbol{w}_t, t). \tag{4}$$

3 Related Efforts

Early work showed that neural networks can learn to generate other neural networks by producing their weights (Ha et al., 2016; Bertinetto et al., 2016; Jia et al., 2016). Building on this, many efforts were made toward modeling the distributions of the weights themselves.

Weight-Space Generative Models. Hyper-Representations and SANE use autoencoders to learn latent representations for weight generation (Schürholt et al., 2022; 2024). P-diff (Wang et al., 2024) combines an autoencoder with latent diffusion, a framework extended with dataset conditioning in D2NWG (Soro et al., 2025). DeepWeightFlow (DWF) operates directly in weight space, with optional PCA compression (Gupta et al., 2026), while RPG scales generation through parameter tokenization and recurrent diffusion (Wang et al., 2025).

Post-Hoc Alignment. Post-hoc approaches align existing checkpoints before training the generator. SANE (Schürholt et al., 2024) and DWF (Gupta et al., 2026) use canonicalization methods such as Git Re-Basin (Ainsworth et al., 2022), which approximately matches weights to a common reference. Erdogan (2025) explores permutation-equivariant FM on MLP trajectory checkpoints, incorporating Git Re-Basin into the training pipeline. For Git Re-Basin, matching assumes equivalent architectures, limiting its direct applicability to heterogeneous collections.

A Priori Alignment. A priori methods artificially reduce permutation variability by building dependent networks. P-diff (Wang et al., 2024) collects checkpoints along an optimization trajectory, a recipe also adopted in D2NWG (Soro et al., 2025). RPG (Wang et al., 2025) similarly collects fine-tuning checkpoints from pretrained models. NNiT (Kim et al., 2026) builds its structurally aligned training collection using a Graph HyperNetwork with a convolutional decoder. Such recipes assume control on creating dependent networks, limiting their applicability to existing repositories of independently trained checkpoints.

Weight-Space Memorization. Zeng et al. (2026) found substantial memorization in P-diff (Wang et al., 2024), Hyper-Representations (Schürholt et al., 2022), HyperDiffusion (Erkoç et al., 2023), and G.pt (Peebles et al., 2022). In the settings studied, they found that these methods largely reproduced training checkpoints or their interpolations and did not outperform simple weight perturbation or averaging baselines in jointly achieving high performance and novelty.

4 METHOD

Our goal is to learn weight distributions from independently trained networks without assuming a common hidden-unit ordering. We motivate our generator through the symmetry of these distributions, then construct a shared equivariant velocity field across architectures and tasks.

Notation. We represent a network by its parameter graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes $\mathcal{V}$, edges $\mathcal{E}$, and $D_{\mathcal{G}}$ parameters. Its vectorized weights and biases are $\boldsymbol{w} \in \mathbb{R}^{D_{\mathcal{G}}}$, while c is a task or domain condition and $\boldsymbol{\theta}$ is the learned parameters of the velocity network. For each architecture, G is the group of function-preserving hidden-unit permutations, acting on $\boldsymbol{w}$ through permutation matrices $\boldsymbol{P}_g$. A probability law p is G-invariant if $\boldsymbol{P}_g \boldsymbol{w}$ has the same law as $\boldsymbol{w} \sim p$ for every $g \in G$. For each architecture $\mathcal{G}$ and task or domain condition c, our target is the conditional weight distribution $p_1(\cdot \mid \mathcal{G}, c)$. We extend the FM velocity field introduced in Section 2 to $v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}_t, t, c)$, sharing its parameters across architectures and tasks while keeping $\mathcal{G}$ and c fixed along each trajectory. We first analyze the flow for a fixed pair $(\mathcal{G}, c)$, suppressing this conditioning in the notation.

4.1 MOTIVATION

We first ask what structure a weight-space generator should respect. Two observations answer this: permutation symmetry is inherited by the distribution of independently trained networks, and consequently by the optimal flow-matching velocity field.

Symmetry of the weight distribution. We first establish an important property of typical weight distributions. For a fixed architecture-task pair $(\mathcal{G}, c)$, consider a training procedure r specifying the dataset, loss, optimizer, hyperparameters, and stopping rule. Let $T_r : \mathbb{R}^{D_{\mathcal{G}}} \to \mathbb{R}^{D_{\mathcal{G}}}$ map an initialization to its final checkpoint. An initialization distribution p_{init} then induces a weight distribution p_1 through $\boldsymbol{w}_1 = T_r(\boldsymbol{w}_{\text{init}})$, where $\boldsymbol{w}_{\text{init}} \sim p_{\text{init}}$. If p_{init} is G-invariant and T_r is equivariant

$$T_r(\boldsymbol{P}_g \boldsymbol{w}_{\text{init}}) = \boldsymbol{P}_g T_r(\boldsymbol{w}_{\text{init}}), \qquad \forall g \in G, \tag{5}$$

then p_1 is G-invariant: permuting trained weights leaves the probability law unchanged. Usual layerwise initializations satisfy this assumption. SGD and Adam respect permutation symmetries when per-step losses and stopping rules are permutation-invariant. The result also holds for mixtures of training recipes, provided each induces a G-invariant distribution. This invariance concerns the underlying weight distribution: a finite training collection need not itself be invariant (Appendix A.1).

Equivariance of the velocity field. Building on the equivariant-flow framework of Köhler et al. (2020), we motivate our parameterization through the symmetry of the optimal flow-matching velocity field. The isotropic Gaussian prior $p_0 = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{D_{\mathcal{G}}})$ is G-invariant, as is the target weight distribution p_1 under the assumptions above. When both prior and target are G-invariant, and under independent draws $\boldsymbol{w}_0 \sim p_0, \boldsymbol{w}_1 \sim p_1$ with a linear interpolation $\boldsymbol{w}_t = (1-t)\boldsymbol{w}_0 + t\boldsymbol{w}_1$, the marginal velocity field $u_t(\boldsymbol{w}) = \mathbb{E}[\boldsymbol{w}_1 - \boldsymbol{w}_0 \mid \boldsymbol{w}_t = \boldsymbol{w}]$ that minimizes the FM objective 3 (Lipman et al., 2023) is almost surely G-equivariant (Appendix A.2):

$$u_t(\boldsymbol{P}_g \boldsymbol{w}) = \boldsymbol{P}_g u_t(\boldsymbol{w}), \qquad \forall g \in G, \quad \forall t \in [0,1]. \tag{6}$$

Since under natural conditions the optimal velocity field is equivariant, restricting the hypothesis class to equivariant fields is a justified inductive bias for learning from finite training collections. Equivariant parameterization also nullifies training-collection alignment (see Section 4.2).

4.2 EQUIVARIANT WEIGHT-SPACE FLOW MATCHING

We now parameterize the shared conditional velocity field $v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}_t, t, c)$ to satisfy the equivariance in Equation 6 for each fixed $(\mathcal{G}, c)$.

Graph representation. Following Lim et al. (2024) and Kofinas et al. (2024), we encode the network parameters on a graph. Shared encoders map node positional and layer information into features $\boldsymbol{h}_i^{(0)}$. Edge features $\boldsymbol{e}_{ij}^{(0)}$ combine encoded weights with embeddings of the edge's positional information, layer, and parameter type. We also maintain a global feature $\boldsymbol{u}^{(\ell)}$ for the graph at each layer ℓ. Positional encodings respect hidden-unit permutations: interchangeable neurons within a layer share an encoding, while fixed roles, such as input coordinates and output classes, do not.

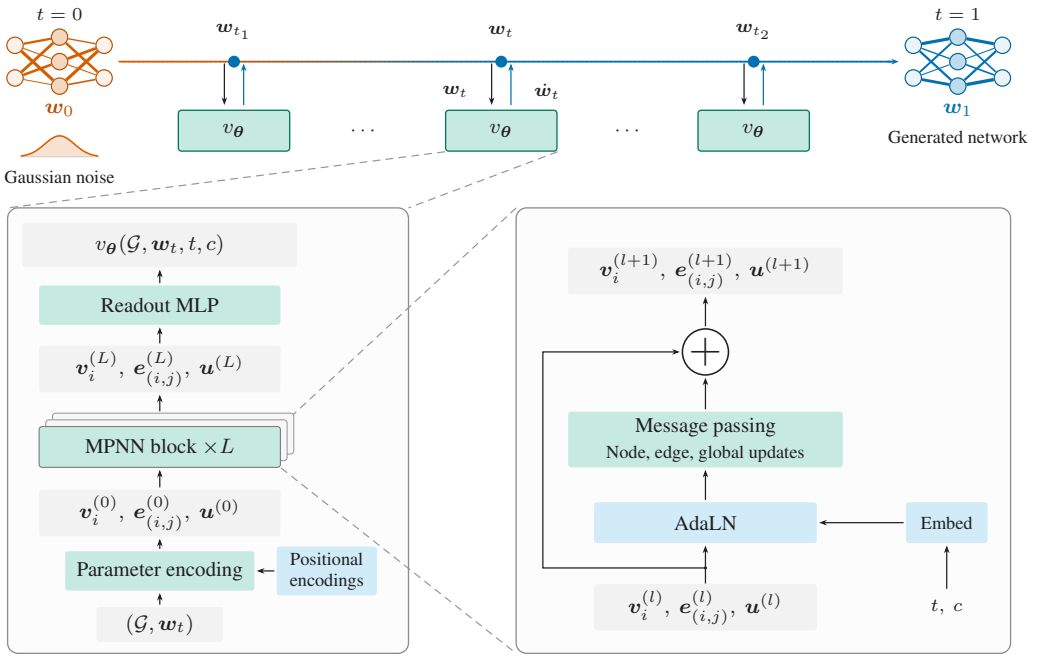

Figure 3: **Equivariant velocity network.** Top: an ODE solver transforms Gaussian weights into a functional network through repeated velocity evaluations. Bottom left: graph encoding, L message-passing blocks, and parameter-velocity prediction. Bottom right: each block combines time- and task-conditioned AdaLN, message passing, and residual updates.

Architecture. We parameterize $v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}_t, t, c)$ with a message-passing GMN over the parameter graph $\mathcal{G}$ (Figure 3). The GMN uses L message-passing blocks. Let $\boldsymbol{h}_i^{(\ell)}$, $\boldsymbol{e}_{ij}^{(\ell)}$, and $\boldsymbol{u}^{(\ell)}$ denote the features of node i, edge (i,j), and the graph as a whole at layer ℓ, respectively.

Time t and condition c enter each block via Adaptive Layer Normalization (AdaLN) (Peebles & Xie, 2023) with joint embedding $\boldsymbol{z}(t,c)$. Bars, e.g., $\bar{\boldsymbol{h}}$, denote modulated features. Let $\mathcal{N}(i)$ be the neighbors of node i and $\bigoplus$ a permutation-invariant aggregator. For $0 \le \ell \le L-1$, the updates are:

$$\left(\bar{\boldsymbol{h}}_i^{(\ell)}, \bar{\boldsymbol{e}}_{ij}^{(\ell)}, \bar{\boldsymbol{u}}^{(\ell)}\right) = \mathrm{AdaLN}_\ell\left(\left(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{e}_{ij}^{(\ell)}, \boldsymbol{u}^{(\ell)}\right), \boldsymbol{z}(t,c)\right), \tag{7}$$

$$\boldsymbol{h}_i^{(\ell+1)} = \boldsymbol{h}_i^{(\ell)} + \mathrm{MLP}_{\mathrm{node}}^{(\ell)}\left(\bar{\boldsymbol{h}}_i^{(\ell)}, \bigoplus_{j \in \mathcal{N}(i)} \mathrm{MLP}_{\mathrm{msg}}^{(\ell)}\left(\bar{\boldsymbol{h}}_i^{(\ell)}, \bar{\boldsymbol{h}}_j^{(\ell)}, \bar{\boldsymbol{e}}_{ij}^{(\ell)}, \bar{\boldsymbol{u}}^{(\ell)}\right), \bar{\boldsymbol{u}}^{(\ell)}\right), \tag{8}$$

$$\boldsymbol{e}_{ij}^{(\ell+1)} = \boldsymbol{e}_{ij}^{(\ell)} + \mathrm{MLP}_{\mathrm{edge}}^{(\ell)}\left(\bar{\boldsymbol{h}}_i^{(\ell)}, \bar{\boldsymbol{h}}_j^{(\ell)}, \bar{\boldsymbol{e}}_{ij}^{(\ell)}, \bar{\boldsymbol{u}}^{(\ell)}\right), \tag{9}$$

$$\boldsymbol{u}^{(\ell+1)} = \boldsymbol{u}^{(\ell)} + \mathrm{MLP}_{\mathrm{global}}^{(\ell)}\left(\bigoplus_{i \in \mathcal{V}} \bar{\boldsymbol{h}}_i^{(\ell)}, \bigoplus_{(i,j) \in \mathcal{E}} \bar{\boldsymbol{e}}_{ij}^{(\ell)}, \bar{\boldsymbol{u}}^{(\ell)}\right). \tag{10}$$

A shared readout MLP uses the final edge feature $\boldsymbol{e}_{ij}^{(L)}$ to predict the velocity of each weight. The resulting field is equivariant to hidden-unit permutations (Appendix A.3).

Training. For each architecture-task pair $(\mathcal{G}, c)$, $p_{\mathrm{data}}(\cdot \mid \mathcal{G}, c)$ is its empirical weight distribution. Drawing independent $\boldsymbol{w}_0 \sim p_0^{(\mathcal{G})} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{D_\mathcal{G}})$, $\boldsymbol{w}_1 \sim p_{\mathrm{data}}(\cdot \mid \mathcal{G}, c)$, and $t \sim \mathcal{U}(0,1)$, we minimize

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathcal{G},c),\, t,\, \boldsymbol{w}_0,\, \boldsymbol{w}_1}\left[\left\| v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}_t, t, c) - (\boldsymbol{w}_1 - \boldsymbol{w}_0)\right\|_2^2\right], \tag{11}$$

with $\boldsymbol{w}_t = (1-t)\boldsymbol{w}_0 + t\boldsymbol{w}_1$. The expectation averages over the training architecture-task pairs. For sampling we use classifier-free guidance (CFG) (Ho & Salimans, 2022), with SDEdit-inspired refinement (Meng et al., 2022) where specified, alternating Gaussian corruption with flow integration.

Alignment leaves the expected objective unchanged. For a permutation-equivariant velocity field, an invariant prior, and independent coupling, applying a separate hidden-unit permutation to each training network leaves the expected flow-matching objective unchanged. In particular, each target permutation can be absorbed into a corresponding permutation of the noise, whose distribution is unchanged (see proof in Appendix A.4). Our model is therefore indifferent to how the training networks are permuted: alignment preprocessing can neither help nor hurt it, so it can be omitted rather than approximated at a cost (Table 2).

4.3 Evaluating Generation Beyond Memorization

To assess generation beyond checkpoint memorization (Zeng et al., 2026), we compare joint task-performance, functional-similarity, and weight-similarity statistics of generated and independently trained networks. High performance can reflect copying, while low similarity can reflect nonfunctional outputs (Figure 4). For a fixed task and architecture, $\mathcal{T} = \{\boldsymbol{w}_i\}_{i=1}^{N}$ is the training collection.

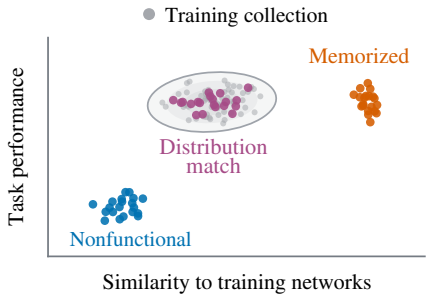

Figure 4: **Generation joint evaluation (conceptual).** Memorized networks perform well; nonfunctional networks have low similarity. We seek agreement with training networks in both metrics.

Task performance and functional similarity. We use test accuracy, or positive-class ROC AUC, as the task-performance score $S_{\text{task}}(\boldsymbol{w})$, and maximum error-IoU as functional similarity $S_{\text{func}}(\boldsymbol{w}; \mathcal{T})$, which is a reference measure of error-pattern similarity in weight-space generation (Zeng et al., 2026; Wang et al., 2024; Gupta et al., 2026). See Appendix B for a definition.

Weight similarity. Functional similarity may be fooled by small training weight perturbations. We thus also measure matched Weight Cosine Similarity (WCS). Let $\boldsymbol{P}_i$ be the hidden-unit permutation found by aligning $\boldsymbol{w}_i$ to $\boldsymbol{w}$ using Git Re-Basin (weight variant). We define

$$\text{WCS}(\boldsymbol{w}; \mathcal{T}) = \max_{\boldsymbol{w}_i \in \mathcal{T}} \frac{\boldsymbol{w}^\top \boldsymbol{P}_i \boldsymbol{w}_i}{\|\boldsymbol{w}\|_2 \|\boldsymbol{P}_i \boldsymbol{w}_i\|_2}. \tag{12}$$

Joint Wasserstein Similarity. Separate average scores mask how performance and similarity vary together. Instead, we combine for each network the three scores: task performance and functional and weight similarities. We compare generated networks with those from the training collection, excluding self-comparisons when computing the latter's similarity scores. We scale the three scores to balance their contributions, applying the same factors from the fixed training reference to generated and training networks. Let $\{\boldsymbol{s}_i\}_{i=1}^{M}$ be the resulting score vectors of M generated networks and $\{\boldsymbol{r}_j^{(b)}\}_{j=1}^{M}$ those of the b-th equally sized sampled subset of the training reference. To compare these sets of scores, we use the Wasserstein-2 distance, which measures the cost to match their elements,

$$W_2^{(b)} = \left[\min_{g \in S_M} \frac{1}{M} \sum_{i=1}^{M} \left\| \boldsymbol{s}_i - \boldsymbol{r}_{g(i)}^{(b)} \right\|_2^2 \right]^{1/2}, \tag{13}$$

S_M being the permutation group on M indices. We introduce Joint Wasserstein Similarity (JWS) as

$$\text{JWS} = \max \left\{ 0,\ 1 - \frac{1}{B\sqrt{3}} \sum_{b=1}^{B} W_2^{(b)} \right\}. \tag{14}$$

Division by B averages over reference samples, while division by $\sqrt{3}$ gives a root-mean-square difference per coordinate. Higher scores indicate closer agreement in the joint performance and similarity statistics. This is our main quantitative generation score. More details in Appendix B.

5 Experiments

We evaluate unconditional and conditional generation on collections of up to 100K *independently* trained image classifiers and tabular predictors. Throughout, we compare against generators that rely on alignment or on dependent training collections with emphasis on assessing memorization.

Table 1: Unconditional generation on MNIST and CIFAR-10. Accuracy (%), Max IoU, and WCS report mean$_{\pm \text{std}}$ across networks; JWS (↑) averages over reference subsamples. Reference JWS $= 1$ denotes an identity comparison. † indicates aligned training weights. Perturbation reports the highest JWS over tested noise levels ($\sigma = 0.25$ for MNIST; 0.10 for CIFAR-10). GMN results include flow-guided refinement. Bold marks the highest JWS among evaluated methods.

Model	**MNIST**				**CIFAR-10**			
	Acc. (%)	**IoU**	**WCS**	**JWS (↑)**	**Acc. (%)**	**IoU**	**WCS**	**JWS (↑)**
Training ref.	$95.9_{\pm 0.2}$	$0.559_{\pm 0.015}$	$0.557_{\pm 0.010}$	1.00	$79.6_{\pm 0.4}$	$0.609_{\pm 0.006}$	$0.609_{\pm 0.013}$	1.00
Random init.	$10.1_{\pm 2.4}$	$0.052_{\pm 0.002}$	$0.138_{\pm 0.003}$	0.15	$10.1_{\pm 1.7}$	$0.254_{\pm 0.012}$	$0.255_{\pm 0.006}$	0.31
Perturbation	$92.2_{\pm 2.2}$	$0.403_{\pm 0.073}$	$0.971_{\pm 0.001}$	0.54	$74.8_{\pm 1.7}$	$0.610_{\pm 0.047}$	$0.995_{\pm 0.000}$	0.63
MLP	$9.8_{\pm 3.7}$	$0.052_{\pm 0.002}$	$0.355_{\pm 0.009}$	0.23	$10.0_{\pm 1.6}$	$0.256_{\pm 0.012}$	$0.146_{\pm 0.003}$	0.25
DiT	$72.0_{\pm 6.5}$	$0.138_{\pm 0.032}$	$0.555_{\pm 0.010}$	0.54	$43.7_{\pm 5.5}$	$0.326_{\pm 0.025}$	$0.553_{\pm 0.015}$	0.62
P-diff	$9.9_{\pm 0.6}$	$0.052_{\pm 0.000}$	$0.404_{\pm 0.009}$	0.25	$10.0_{\pm 0.1}$	$0.249_{\pm 0.000}$	$0.234_{\pm 0.004}$	0.29
DWF	$10.9_{\pm 3.6}$	$0.052_{\pm 0.002}$	$0.531_{\pm 0.009}$	0.27	$10.0_{\pm 0.1}$	$0.249_{\pm 0.001}$	$0.432_{\pm 0.009}$	0.37
DWF†	$72.9_{\pm 6.5}$	$0.145_{\pm 0.034}$	$0.595_{\pm 0.015}$	0.55	$39.6_{\pm 6.6}$	$0.315_{\pm 0.024}$	$0.671_{\pm 0.019}$	0.59
SANE$^{\dagger}_{\text{Gauss}}$	$16.9_{\pm 5.5}$	$0.054_{\pm 0.003}$	$0.531_{\pm 0.018}$	0.29	$10.9_{\pm 1.9}$	$0.255_{\pm 0.009}$	$0.482_{\pm 0.014}$	0.39
SANE$^{\dagger}_{\text{KDE}}$	$93.9_{\pm 1.6}$	$0.470_{\pm 0.066}$	$0.941_{\pm 0.004}$	0.59	$72.9_{\pm 2.5}$	$0.555_{\pm 0.054}$	$0.982_{\pm 0.002}$	0.64
GMN (Ours)	$94.4_{\pm 0.4}$	$0.469_{\pm 0.022}$	$0.553_{\pm 0.008}$	**0.91**	$75.3_{\pm 0.8}$	$0.533_{\pm 0.012}$	$0.630_{\pm 0.011}$	**0.92**

5.1 Unconditional Generation

Datasets. We use unaligned collections of 10K independently trained MNIST MLP classifiers with two hidden layers (LeCun et al., 1998) and 50K CIFAR-10 CNN classifiers with three convolutional layers and a linear classifier (Krizhevsky, 2009). All networks use distinct initialization seeds.

Baselines. We evaluate DWF (Gupta et al., 2026), P-diff (Wang et al., 2024), SANE (Schürholt et al., 2024), and MLP and DiT (Peebles & Xie, 2023) velocity networks. DWF is tested with and without Git Re-Basin alignment. SANE uses aligned weights, permutation augmentation, and KDE or Gaussian sampling. Remaining baselines use unaligned weights. Random weights and perturbed training checkpoints serve as controls. Experimental details and baseline sweeps are in Appendix C.

Results. Table 1 reports metrics from Section 4.3. MLP, P-diff, and unaligned DWF are near chance; alignment helps DWF but accuracy stays below reference. SANE's Gaussian mode performs poorly; KDE sampling and perturbation retain accuracy despite excessive weight similarity, highlighting why measuring accuracy alone is insufficient. Our GMN achieves the closest joint match: JWS of 0.91/0.92, versus at most 0.59/0.64 for baselines.

Equivariance and Alignment. The objective invariance in Section 4.2 motivates training without alignment. Retraining the MNIST GMN on the same collection after Git Re-Basin alignment yields nearly identical accuracy and similarity statistics (Table 2).

Table 2: The equivariant GMN trained on the same MNIST collection, unaligned (numbers from Table 1) and aligned via Git Re-Basin. Notation as in Table 1.

Alignment	**Acc. (%)**	**IoU**	**WCS**	**JWS (↑)**
Unaligned	$94.4_{\pm 0.4}$	$0.469_{\pm 0.022}$	$0.553_{\pm 0.008}$	0.91
Aligned	$94.3_{\pm 0.6}$	$0.464_{\pm 0.023}$	$0.552_{\pm 0.008}$	0.90

5.1.1 Sample and Model Complexity Analysis

Recent studies ask whether scale can substitute for symmetry (Brehmer et al., 2024). We investigate this question in weight-space generation by varying training collection size and model capacity.

Dataset and setup. We vary the training collection from 100 to 50K independently trained MLPs on `mfeat-karhunen` (Bischl et al., 2021), comparing our 2M-parameter GMN with 2M and 10M DWF and DiT models. Both baselines operate directly on weights without dimensionality reduction.

Results. Figure 5 shows that high baseline accuracy at small collection sizes coincides with near-unit Max-IoU and WCS, indicating memorization. Our 2M GMN achieves the highest JWS throughout the sweep, approaching the reference performance and similarity statistics as the collection grows. Within the evaluated training budgets, increasing baseline capacity fivefold or applying alignment does not close the gap in joint generation quality. Technical details appear in Appendix D.

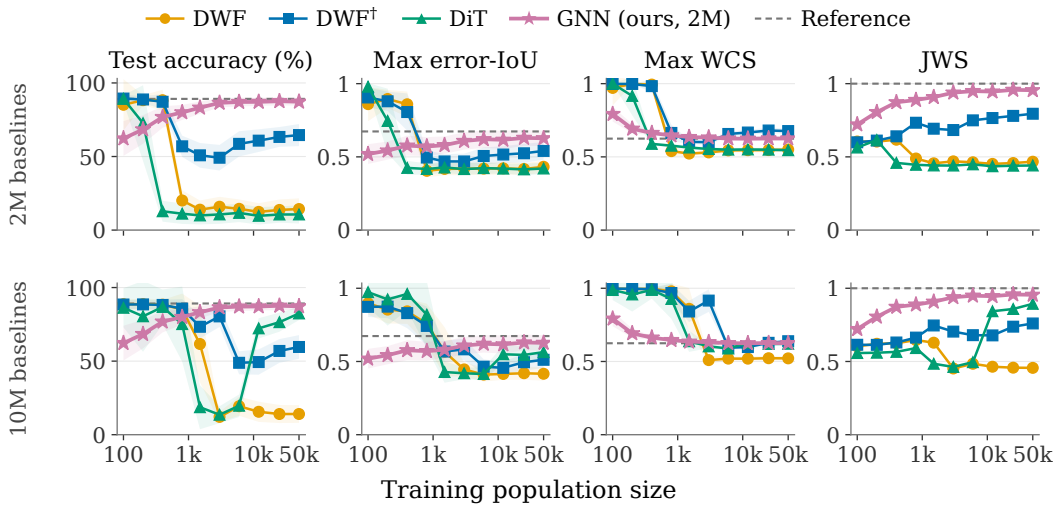

Figure 5: **Collection and capacity scaling on `mfeat-karhunen`.** Baselines have 2M (top) or 10M (bottom) parameters. Columns report test accuracy, Max error-IoU, Max WCS, and JWS. Dashed lines are collection reference means in the first three columns and JWS $= 1$ in the last. Bands show ± 1 standard deviation. † denotes post-hoc alignment via Git Re-Basin.

5.2 Conditional Generation

We next test whether a shared velocity field can generate networks for different tasks and architectures and produce useful models at domain conditions absent from training.

5.2.1 Heterogeneous Architecture Generation and Unseen Width Transfer

Datasets and setup. Our framework is not limited to networks of the same architecture. We create an (inter) unalignable collection of 100K independently trained MLPs on 20 tabular datasets from OpenML (Bischl et al., 2021), with 5K networks per dataset. The collection spans 16 architectures, varying in input dimensionality, output classes, depth, and hidden widths. Four pairs of tasks share the same architecture, so architecture alone does not identify the task. We train a single conditional GNN flow, supplying the task identity c separately from the parameter graph $\mathcal{G}$. All architectures share the velocity network's parameters, and none are architecture-specific.

Results. Our generated networks achieve a mean test accuracy of 82.17%, compared with the collection's 83.15% (Figure 6). Remarkably, our method can even zero-shot generalize to new architectures it has never seen. We test this by scaling every hidden width by $0.5\times$ or $1.5\times$. Only the architecture changes; our velocity GMN requires neither new parameters nor retraining. We get a mean accuracy of 78.96% for narrow and 82.35% for wide networks. Performance varies across tasks: `letter` has the largest gap to its original collection reference, while width reduction most strongly affects `letter` and `mfeat-karhunen`. Details are provided in Appendix F.

5.2.2 Domain-Conditioned Generation

Dataset and setup. Classifiers may fail outside their training domain. We test if conditional generation yields individual classifiers accurate across domains. On Folktables/ACSIncome (Ding et al., 2021) data from California (CA) and Puerto Rico (PR), exhibiting domain shift as Californians tend to be wealthier, we independently train 5K classifiers per domain to predict if annual income exceeds \$50K. We train a conditional flow on their weights only at $c = 0$ (CA) and $c = 1$ (PR). We sample at both endpoints and unseen $c \in (0, 1)$. Using ROC AUC on both domains, we compare generated networks with CA–PR weight averages (with and without Git Re-Basin alignment) and oracle logit ensembles. We evaluate JWS against each domain's reference cloud. Details in Appendix E.

Results. Endpoint conditioning yields networks with domain-specific performance profiles (Figure 7). Remarkably, at intermediate conditions absent from training, generated networks move

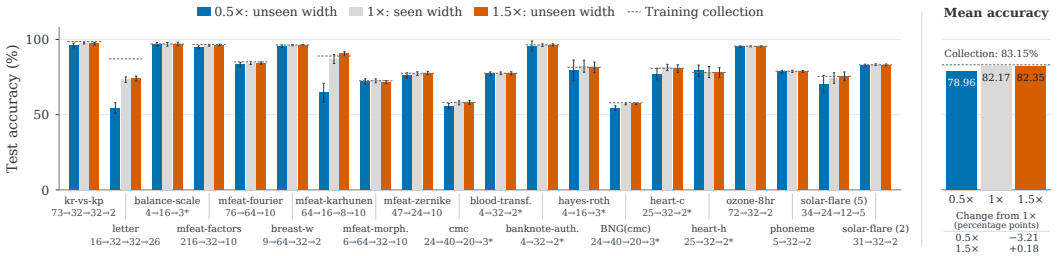

Figure 6: **Heterogeneous generation and zero-shot width transfer.** A single velocity network with shared parameters generates networks across all 20 tasks. Base architectures list layer widths from input to output; $*$ marks architectures shared by two tasks. Narrow and Wide multiply all hidden widths by $0.5\times$ and $1.5\times$, respectively, while preserving input and output dimensions. Reference reports the mean accuracy of independently trained networks at the base architecture. Generated accuracies are mean$_{\pm\text{std}}$ (%) over 100 networks per entry. The final plot averages the task means.

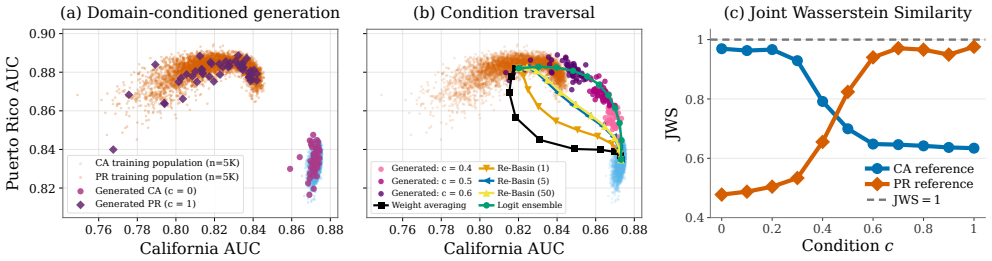

Figure 7: **ACS domain conditioning.** Faint points show training collections. **(a)** Endpoint generation. **(b)** Intermediate conditions $c \in \{0.4, 0.5, 0.6\}$ and AUC comparison curves. Panels (a,b) use 50 networks per condition, with guidance scales 1.5 and 4.0, respectively. Comparison curves average 50 CA–PR checkpoint pairs; Re-Basin labels denote matching iterations. **(c)** JWS against both domain references, using 100 generated networks at each $c \in \{0, 0.1, \ldots, 1\}$.

along the CA–PR performance trade-off, achieving cross-domain AUC comparable to two-network logit ensembles. Unaligned weight averaging degrades performance; post-hoc alignment improves the mixtures. As a complementary check, JWS decreases from $0.969/0.976$ at the endpoints to $0.700/0.824$ at $c = 0.5$ (CA/PR), consistent with a shift away from domain reference collections.

6 Conclusion

We proposed a permutation-equivariant GMN velocity field for learning weight distributions directly from independently trained networks, without alignment or single-trajectory construction. Experiments show our accurate generation with joint performance and similarity statistics closer to independently trained networks than with any other baseline. A single conditional model generates weights across heterogeneous architectures, transfers to unseen hidden widths, and yields individual classifiers with cross-domain performance comparable to logit ensembles. These results support incorporating weight-space permutation symmetries directly into generative architectures.

Limitations and future work. Our experiments focus on fairly small networks as a first step to show accurate weight-space generation. Regardless of scalability, generation should first work at small scales without memorization. Though we believe results would hold on larger models, scaling to them is left to future work as dense graph representations cost substantial memory and compute, potentially reduced with custom GPU kernels and optimized sparse operations (Hu et al., 2020). While graph representations extend to Transformers (Lim et al., 2024), generating foundation-model weights remains an open challenge. Beyond scaling, conditioning currently uses task identifiers rather than dataset features, and we do not evaluate generation on unseen tasks. Conditioning on dataset representations, for example through Set Transformers (Lee et al., 2019) or DeepSets (Zaheer et al., 2017), remains an open direction. The JWS score jointly evaluates performance and similarity, yet complementary metrics would be welcome to more broadly assess weight space generation. Nevertheless, our work opens exciting avenues to explore in weight space generation.

AI USE STATEMENT

In this work, we used generative AI tools for text polishing and assistance in creating figures. No AI was used to develop the theoretical framework, generate research ideas, or design experiments.

REPRODUCIBILITY STATEMENT

Sections 4, 5, B, C, D, E, F provide all the necessary details to reproduce our work, including training hyperparameters, GMN construction, baseline sweeps, and metrics definitions with validation. Code will be provided upon acceptance.

REFERENCES

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

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.

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. *Advances in neural information processing systems*, 29, 2016.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=OCrD8ycKjG`.

Johann Brehmer, Sönke Behrends, Pim De Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

Adir Dayan, Yam Eitan, and Haggai Maron. On the expressive power of permutation-equivariant weight-space networks. *arXiv preprint arXiv:2602.01083*, 2026.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in neural information processing systems*, 34:6478–6490, 2021.

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

Ege Erdogan. Geometric flow models over neural network weights. *arXiv preprint arXiv:2504.03710*, 2025.

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

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, 2025.

Saumya Gupta, Scott Biggs, Moritz Laber, Zohair Shafi, Robin Walters, and Ayan Paul. Deepweightflow: Re-basined flow matching for generating neural network weights. *arXiv preprint arXiv:2601.05052*, 2026.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

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.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. Elsevier, 1990.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Yuwei Hu, Zihao Ye, Minjie Wang, Jiali Yu, Da Zheng, Mu Li, Zheng Zhang, Zhiru Zhang, and Yida Wang. Featgraph: A flexible and efficient backend for graph neural network systems. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–13. IEEE, 2020.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.

Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. *Advances in Neural Information Processing Systems*, 2024.

Jiwoo Kim, Swarajh Mehta, Hao-Lun Hsu, Hyunwoo Ryu, Yudong Liu, and Miroslav Pajic. Nnit: Width-agnostic neural network generation with structurally aligned weight spaces. *arXiv preprint arXiv:2603.00180*, 2026.

Boris Knyazev, Abhinav Moudgil, Guillaume Lajoie, Eugene Belilovsky, and Simon Lacoste-Julien. Accelerating training with neuron interaction and nowcasting networks. In *International Conference on Learning Representations*, volume 2025, pp. 29569–29588, 2025.

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

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International conference on machine learning*, pp. 5361–5370. PMLR, 2020.

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

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

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

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`.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

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

Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. In *International Conference on Machine Learning*, pp. 37376–37395. PMLR, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022.

Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024.

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.

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

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

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

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

Viet-Hoang Tran, Thieu Vo, An Nguyen, Tho-Huu Tran, Minh-Khoi Nguyen-Nhat, Thanh Tran, Duy-Tung Pham, and Tan Nguyen. Equivariant neural functional networks for transformers. In *International Conference on Learning Representations*, volume 2025, pp. 16270–16341, 2025.

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

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

Kai Wang, Dongwen Tang, Wangbo Zhao, Konstantin Schürholt, Zhangyang Wang, and Yang You. Scaling up parameter generation: A recurrent diffusion approach. *Advances in Neural Information Processing Systems*, 38:93014–93046, 2025.

Zhangyang Wang, Peihao Wang, and Kai Wang. Position: Weight space should be a first-class generative ai modality. *arXiv preprint arXiv:2605.18632*, 2026.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Boya Zeng, Yida Yin, Zhiqiu Xu, and Zhuang Liu. Generative modeling of weights: Generalization or memorization? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41974–41984, 2026.

APPENDIX CONTENTS

A Theoretical Details **14**

A.1 Invariance of the Weight Distribution 14

A.2 Equivariance of the Marginal Velocity Field 15

A.3 Equivariance of the GMN Velocity Field 15

A.4 Invariance of the Flow Matching Objective under Alignment 16

B Evaluation Metrics **16**

B.1 Per-Network Scores 16

B.2 Training-Collection Statistics 17

B.3 Generated-Network Statistics 17

B.4 Joint Wasserstein Similarity 17

B.5 Validation of JWS 18

C Unconditional Generation **19**

C.1 Data and Training Collections 19

C.2 GMN Configuration 20

C.3 Parameter-Group Normalization 20

C.4 Sampling and Refinement 21

C.5 Evaluation 21

C.6 Diversity of Generated Networks 22

C.7 Baseline Configurations and Sweeps 22

D Sample and Model Complexity **26**

E Domain-Conditioned Generation **29**

F Heterogeneous Architecture Generation **29**

F.1 Data and Training Collections 29

F.2 Shared Representation and Conditioning 30

F.3 Generation and Width Transfer 31

A THEORETICAL DETAILS

A.1 INVARIANCE OF THE WEIGHT DISTRIBUTION

We restate the weight distribution invariance claim. Fix an architecture $\mathcal{G}$ and a training procedure r. Let $T_r : \mathbb{R}^{D_\mathcal{G}} \to \mathbb{R}^{D_\mathcal{G}}$ be a map from an initialization to its final checkpoint. An initialization distribution p_{init} induces a weight distribution p_1 through $\boldsymbol{w}_1 = T_r(\boldsymbol{w}_{\text{init}})$, where $\boldsymbol{w}_{\text{init}} \sim p_{\text{init}}$. Suppose that p_{init} is G-invariant and T_r is G-equivariant:

$$T_r(\boldsymbol{P}_g \boldsymbol{w}_{\text{init}}) = \boldsymbol{P}_g T_r(\boldsymbol{w}_{\text{init}}), \qquad \forall g \in G. \tag{15}$$

Then p_1 is G-invariant.

Proof. Let $g \in G$ and let $A \subseteq \mathbb{R}^{D_\mathcal{G}}$ be measurable. Then

$$p_1(\boldsymbol{P}_g A) = \Pr(T_r(\boldsymbol{w}_{\text{init}}) \in \boldsymbol{P}_g A) \tag{16}$$
$$= \Pr\big(\boldsymbol{P}_g^{-1} T_r(\boldsymbol{w}_{\text{init}}) \in A\big) \tag{17}$$
$$= \Pr\big(T_r(\boldsymbol{P}_g^{-1}\boldsymbol{w}_{\text{init}}) \in A\big) \tag{18}$$
$$= \Pr(T_r(\boldsymbol{w}_{\text{init}}) \in A) = p_1(A). \tag{19}$$

The third equality follows from equivariance of T_r, and the fourth from invariance of p_{init}. □

Remark 1 (Equivariance of training). Let $\boldsymbol{P}_g$ denote a weight-space symmetry induced by consistently relabeling hidden units or channels, rather than an arbitrary permutation of parameter coordinates. For a differentiable loss invariant under these symmetries, $f(\boldsymbol{P}_g \boldsymbol{w}) = f(\boldsymbol{w})$, the chain rule gives

$$\nabla f(\boldsymbol{P}_g \boldsymbol{w}) = \boldsymbol{P}_g \nabla f(\boldsymbol{w}). \tag{20}$$

As an example, for the toy loss $f(w_1, w_2) = w_1^2 + w_2^2$, swapping $(1, 2)$ to $(2, 1)$ swaps the gradient from $(2, 4)$ to $(4, 2)$.

Gradient descent therefore commutes with hidden-unit permutations. This also holds for SGD and Adam when minibatch losses are permutation-invariant, optimizer states are initialized at zero, hyperparameters are shared across interchangeable coordinates, and the stopping rule is permutation-invariant.

Remark 2 (Index-dependent structure). The claim applies to any group G under which training is equivariant. Operations tied to specific neuron indices may restrict this group. For example, a fixed dropout mask preserves only permutations that leave its pattern unchanged.

Remark 3 (Heterogeneous recipes and conditioning). A mixture of G-invariant weight distributions is also G-invariant. Thus, training recipes may vary across networks, provided each recipe induces a G-invariant distribution. For conditional generation, the same argument applies separately to each architecture-task pair $(\mathcal{G}, c)$, using the corresponding architecture's permutation group.

A.2 Equivariance of the Marginal Velocity Field

Fix an architecture-task pair $(\mathcal{G}, c)$. Let $\boldsymbol{w}_0 \sim p_0$ and $\boldsymbol{w}_1 \sim p_1$ be independent, with G-invariant laws and finite second moments. For $\boldsymbol{w}_t = (1-t)\boldsymbol{w}_0 + t\boldsymbol{w}_1$, the population flow-matching loss at time t is

$$\mathcal{L}_t(v) = \mathbb{E}_{\boldsymbol{w}_0, \boldsymbol{w}_1}\left[\|v(\boldsymbol{w}_t) - (\boldsymbol{w}_1 - \boldsymbol{w}_0)\|_2^2\right]. \tag{21}$$

Its minimizer over vector fields is the marginal velocity field (Lipman et al., 2023),

$$u_t(\boldsymbol{w}) = \mathbb{E}[\boldsymbol{w}_1 - \boldsymbol{w}_0 \mid \boldsymbol{w}_t = \boldsymbol{w}]. \tag{22}$$

We show that $u_t(\boldsymbol{P}_g \boldsymbol{w}) = \boldsymbol{P}_g u_t(\boldsymbol{w})$ for every $g \in G$ and almost every $\boldsymbol{w}$.

Proof. Write $\boldsymbol{\Delta} = \boldsymbol{w}_1 - \boldsymbol{w}_0$. Independence, invariance, and linearity of interpolation imply

$$(\boldsymbol{P}_g \boldsymbol{w}_t, \boldsymbol{P}_g \boldsymbol{\Delta}) \overset{d}{=} (\boldsymbol{w}_t, \boldsymbol{\Delta}).$$

Thus, using this equality in law and invertibility of $\boldsymbol{P}_g$,

$$u_t(\boldsymbol{w}) = \mathbb{E}[\boldsymbol{P}_g \boldsymbol{\Delta} \mid \boldsymbol{P}_g \boldsymbol{w}_t = \boldsymbol{w}] \tag{23}$$
$$= \boldsymbol{P}_g \mathbb{E}[\boldsymbol{\Delta} \mid \boldsymbol{w}_t = \boldsymbol{P}_g^{-1}\boldsymbol{w}] \tag{24}$$
$$= \boldsymbol{P}_g u_t(\boldsymbol{P}_g^{-1}\boldsymbol{w}). \tag{25}$$

Replacing $\boldsymbol{w}$ with $\boldsymbol{P}_g \boldsymbol{w}$ gives the result. □

A.3 Equivariance of the GMN Velocity Field

The construction in Section 4.2 satisfies

$$v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{P}_g \boldsymbol{w}, t, c) = \boldsymbol{P}_g v_{\boldsymbol{\theta}}(\mathcal{G}, \boldsymbol{w}, t, c), \qquad g \in G. \tag{26}$$

Proof. We apply the graph-metanetwork equivariance argument of Lim et al. (2024, Proposition 2 and Appendix B.6); see also Kofinas et al. (2024, Sections 2.4–3).

Each $g \in G$ preserves the connectivity and structural roles of $\mathcal{G}$. Shared encoders and positional encodings constant on G-orbits therefore produce correspondingly permuted initial node and edge features, while the initial global feature is unchanged.

Shared message-passing updates with permutation-invariant aggregation $\bigoplus$ preserve this transformation: node and edge features are permuted, and global features remain unchanged. AdaLN also commutes with these permutations, since it normalizes each feature vector separately and uses the same conditioning embedding $\boldsymbol{z}(t, c)$ across nodes and edges. Residual connections preserve equivariance as well.

Consequently, after all L blocks, the shared readout produces correspondingly permuted parameter velocities, establishing the claim. □

A.4 Invariance of the Flow Matching Objective under Alignment

Fix an architecture-task pair $(\mathcal{G}, c)$. Suppose $v_{\boldsymbol{\theta}}$ is G-equivariant and the prior p_0 is G-invariant. Under independent sampling $\boldsymbol{w}_0 \sim p_0$ and $\boldsymbol{w}_1 \sim p_1$, replacing each target by $\boldsymbol{P}_{\pi(\boldsymbol{w}_1)}\boldsymbol{w}_1$, for any measurable map $\pi : \mathbb{R}^{D_{\mathcal{G}}} \to G$, leaves the expected flow-matching objective unchanged.

Proof. Write the sample loss as

$$\ell_t(\boldsymbol{a}, \boldsymbol{b}) = \|v_{\boldsymbol{\theta}}(\mathcal{G}, (1-t)\boldsymbol{a} + t\boldsymbol{b}, t, c) - (\boldsymbol{b} - \boldsymbol{a})\|_2^2 .$$

Fix t and $\boldsymbol{w}_1$, and let $\boldsymbol{P} = \boldsymbol{P}_{\pi(\boldsymbol{w}_1)}$. Equivariance and orthogonality of $\boldsymbol{P}$ imply

$$\ell_t(\boldsymbol{w}_0, \boldsymbol{P}\boldsymbol{w}_1) = \ell_t(\boldsymbol{P}^{-1}\boldsymbol{w}_0, \boldsymbol{w}_1).$$

Since $\boldsymbol{w}_0$ is independent of $\boldsymbol{w}_1$, its distribution remains p_0 when the target is fixed. Moreover, invariance of p_0 means that $\boldsymbol{P}^{-1}\boldsymbol{w}_0$ also has distribution p_0. Therefore,

$$\mathbb{E}_{\boldsymbol{w}_0 \sim p_0}[\ell_t(\boldsymbol{w}_0, \boldsymbol{P}\boldsymbol{w}_1)] = \mathbb{E}_{\boldsymbol{w}_0 \sim p_0}[\ell_t(\boldsymbol{P}^{-1}\boldsymbol{w}_0, \boldsymbol{w}_1)] \tag{27}$$

$$= \mathbb{E}_{\boldsymbol{w}_0 \sim p_0}[\ell_t(\boldsymbol{w}_0, \boldsymbol{w}_1)]. \tag{28}$$

This equality holds for every fixed t and $\boldsymbol{w}_1$. Averaging both sides over time and targets gives

$$\mathbb{E}_{t, \boldsymbol{w}_0, \boldsymbol{w}_1}[\ell_t(\boldsymbol{w}_0, \boldsymbol{P}_{\pi(\boldsymbol{w}_1)}\boldsymbol{w}_1)] = \mathbb{E}_{t, \boldsymbol{w}_0, \boldsymbol{w}_1}[\ell_t(\boldsymbol{w}_0, \boldsymbol{w}_1)],$$

which is the claimed invariance of the training objective. □

B Evaluation Metrics

Following Section 4.3, we evaluate task performance, functional similarity, matched weight similarity, and their joint distribution through JWS.

B.1 Per-Network Scores

We consider a fixed task and architecture, with training collection $\mathcal{T}$. Below, $\mathcal{A}$ denotes the collection against which a query network is compared. All functional comparisons use the same test set. For ACS, evaluation is performed separately for CA and PR, using the corresponding training collection and test set.

Task performance. The score $S_{\text{task}}(\boldsymbol{w})$ is test accuracy for MNIST, CIFAR-10, and OpenML tasks, and positive-class ROC AUC for ACS.

Max IoU. Following Zeng et al. (2026); Gupta et al. (2026); Wang et al. (2024), we measure functional similarity through overlap between classification errors. Let $\mathcal{E}_{\boldsymbol{w}}$ denote the indices of test examples misclassified by $\boldsymbol{w}$. We define

$$S_{\text{func}}(\boldsymbol{w}; \mathcal{A}) = \text{MaxIoU}(\boldsymbol{w}; \mathcal{A}) = \max_{\boldsymbol{w}_j \in \mathcal{A}} \frac{|\mathcal{E}_{\boldsymbol{w}} \cap \mathcal{E}_{\boldsymbol{w}_j}|}{|\mathcal{E}_{\boldsymbol{w}} \cup \mathcal{E}_{\boldsymbol{w}_j}|}. \tag{29}$$

Lower values indicate less overlap with the errors of networks in $\mathcal{A}$. Since this score also depends on classification error rates, we interpret it jointly with task performance.

Matched weight cosine similarity. We align each $\boldsymbol{w}_j \in \mathcal{A}$ to $\boldsymbol{w}$ using weight matching in Git Re-Basin (Ainsworth et al., 2022). Writing the resulting permutation as $\boldsymbol{P}_{\boldsymbol{w},j}$, we define

$$\mathrm{WCS}(\boldsymbol{w}; \mathcal{A}) = \max_{\boldsymbol{w}_j \in \mathcal{A}} \frac{\boldsymbol{w}^\top \boldsymbol{P}_{\boldsymbol{w},j} \boldsymbol{w}_j}{\|\boldsymbol{w}\|_2 \|\boldsymbol{P}_{\boldsymbol{w},j} \boldsymbol{w}_j\|_2}. \tag{30}$$

Values near 1 indicate similar weight directions after matching. Since matching is approximate, WCS lower-bounds the maximum cosine similarity over the comparison collection and all valid hidden-unit permutations.

B.2 Training-Collection Statistics

For a training network $\boldsymbol{w}_i \in \mathcal{T}$, we evaluate task performance directly and compute both similarity scores against $\mathcal{T} \setminus \{\boldsymbol{w}_i\}$. This leave-one-out comparison excludes trivial self-matches. The mean task-performance and functional-similarity scores are

$$\overline{S}^{\mathrm{train}}_{\mathrm{task}} = \frac{1}{|\mathcal{T}|} \sum_{\boldsymbol{w}_i \in \mathcal{T}} S_{\mathrm{task}}(\boldsymbol{w}_i), \tag{31}$$

$$\overline{S}^{\mathrm{train}}_{\mathrm{func}} = \frac{1}{|\mathcal{T}|} \sum_{\boldsymbol{w}_i \in \mathcal{T}} S_{\mathrm{func}}(\boldsymbol{w}_i; \mathcal{T} \setminus \{\boldsymbol{w}_i\}). \tag{32}$$

Computing WCS for every training network requires a quadratic number of pairwise comparisons. We therefore estimate its mean using a uniformly sampled subset $\mathcal{R} \subset \mathcal{T}$ of $m = 1{,}000$ reference networks:

$$\widehat{\mu}^{\mathrm{train}}_{\mathrm{WCS}} = \frac{1}{m} \sum_{\boldsymbol{w}_i \in \mathcal{R}} \mathrm{WCS}(\boldsymbol{w}_i; \mathcal{T} \setminus \{\boldsymbol{w}_i\}). \tag{33}$$

Only the query networks are subsampled; each is compared against the remaining full training collection.

B.3 Generated-Network Statistics

For each generated network $\widetilde{\boldsymbol{w}}_k$, we evaluate task performance directly and compute both similarity scores against the full training collection $\mathcal{T}$. For M generated networks, the reported means are

$$\overline{S}^{\mathrm{gen}}_{\mathrm{task}} = \frac{1}{M} \sum_{k=1}^{M} S_{\mathrm{task}}(\widetilde{\boldsymbol{w}}_k), \tag{34}$$

$$\overline{S}^{\mathrm{gen}}_{\mathrm{func}} = \frac{1}{M} \sum_{k=1}^{M} S_{\mathrm{func}}(\widetilde{\boldsymbol{w}}_k; \mathcal{T}), \tag{35}$$

$$\overline{\mathrm{WCS}}^{\mathrm{gen}} = \frac{1}{M} \sum_{k=1}^{M} \mathrm{WCS}(\widetilde{\boldsymbol{w}}_k; \mathcal{T}). \tag{36}$$

We also report the standard deviation of each score across the generated networks.

B.4 Joint Wasserstein Similarity

Separate metric averages can hide mixtures of memorized and poorly performing networks. JWS compares the joint distribution of task performance, functional similarity, and matched weight similarity.

Reference cloud. For a network $\boldsymbol{w}$ and comparison collection $\mathcal{A}$, define

$$\mathbf{v}(\boldsymbol{w}; \mathcal{A}) = \big(S_{\mathrm{task}}(\boldsymbol{w}), S_{\mathrm{func}}(\boldsymbol{w}; \mathcal{A}), \mathrm{WCS}(\boldsymbol{w}; \mathcal{A})\big)^\top. \tag{37}$$

The reference cloud uses the uniformly sampled subset $\mathcal{R} \subset \mathcal{T}$ of $m = 1{,}000$ networks for which all three scores are available. Each reference network $\boldsymbol{w}_j \in \mathcal{R}$ contributes $\mathbf{v}(\boldsymbol{w}_j; \mathcal{T} \setminus \{\boldsymbol{w}_j\})$, whereas each generated network $\widetilde{\boldsymbol{w}}_i$ contributes $\mathbf{v}(\widetilde{\boldsymbol{w}}_i; \mathcal{T})$. Thus, only the reference queries are subsampled; similarity comparisons use the full training collection, excluding self-comparisons for reference networks. All three coordinates of each point belong to the same network. For ACS, we construct a separate reference cloud for each domain.

Coordinate scaling. We compute the coordinate means from these reference vectors and define

$$\mu_k = \frac{1}{m} \sum_{\boldsymbol{w}_j \in \mathcal{R}} v_k(\boldsymbol{w}_j; \mathcal{T} \setminus \{\boldsymbol{w}_j\}), \qquad d_k = \max(\mu_k, 1 - \mu_k). \tag{38}$$

We scale coordinate k by $\widetilde{v}_k = v_k / d_k$. The scale d_k is the distance from μ_k to the farther endpoint of $[0, 1]$, expressing discrepancies relative to this reference-dependent range. The same scales apply to generated and reference networks and remain fixed across reference subsamples and evaluated methods.

Estimation and normalization. As in the main text, let

$$\mathbf{s}_i = \widetilde{\mathbf{v}}(\widetilde{\boldsymbol{w}}_i; \mathcal{T}), \qquad i = 1, \ldots, M, \tag{39}$$

denote the scaled scores of M generated networks. For each $b = 1, \ldots, B$, we sample $M \leq m$ networks uniformly without replacement from $\mathcal{R}$. Writing these networks as $\{\boldsymbol{w}_j^{(b)}\}_{j=1}^{M}$, their scaled scores are

$$\mathbf{r}_j^{(b)} = \widetilde{\mathbf{v}}\big(\boldsymbol{w}_j^{(b)}; \mathcal{T} \setminus \{\boldsymbol{w}_j^{(b)}\}\big). \tag{40}$$

These scores are selected from the fixed reference cloud; they are not recomputed against the sampled subset. We compute

$$W_2^{(b)} = \left[\min_{g \in S_M} \frac{1}{M} \sum_{i=1}^{M} \left\| \mathbf{s}_i - \mathbf{r}_{g(i)}^{(b)} \right\|_2^2 \right]^{1/2}, \tag{41}$$

where S_M is the permutation group on M indices. We solve this one-to-one matching using exact linear assignment with squared-Euclidean costs and report

$$\mathrm{JWS} = \max \left\{ 0,\ 1 - \frac{1}{B\sqrt{3}} \sum_{b=1}^{B} W_2^{(b)} \right\}. \tag{42}$$

Division by B averages over reference subsamples, while division by $\sqrt{3}$ expresses each distance as a root-mean-square discrepancy per coordinate. Clipping keeps JWS in $[0, 1]$. Higher scores indicate closer agreement in the joint performance and similarity statistics. Across experiments, we use $B = 100$ reference subsamples per JWS estimate, with equal generated sample sizes across methods within each experiment.

B.5 Validation of JWS

Sanity check. We test whether JWS assigns high scores to held-out networks trained using the same procedure as the original collections. We train 100 additional MNIST MLPs, 100 CIFAR-10 CNNs, and 1,000 `mfeat-karhunen` MLPs using the original architectures and recipes with new seeds, following Table 4 and Section D. These networks are excluded from generator training and reference construction. We evaluate them against the original training collections, keeping the reference clouds and normalization scales fixed. Their high JWS scores (Table 3(a)) provide a positive control: independently trained, held-out networks closely reproduce the reference joint statistics.

Normalization ablation. Table 3 (b) ablates coordinate scaling and the $\sqrt{3}$ distance normalizer. Without coordinate scaling, the gap between GMN and the strongest baseline decreases from 0.32 to 0.18 on MNIST and from 0.28 to 0.17 on CIFAR-10. Removing $\sqrt{3}$ clips five methods per dataset to zero, obscuring differences between them. GMN ranks highest under all three variants, showing that its advantage persists across these normalization choices.

Table 3: **JWS validation. (a)** Held-out networks use the original architectures and training recipes with disjoint seeds, evaluated using the original reference clouds and scales. M is the number of held-out networks evaluated. **(b)** Full uses both normalizations; Raw removes coordinate scaling; No $\sqrt{3}$ removes only the final distance normalizer. All variants retain clipping at zero. † denotes aligned training weights. Bold marks the highest score among evaluated methods.

(a) Held-out network sanity check

Dataset	**MNIST**	**CIFAR-10**	`mfeat-karhunen`		
M	100	100	100	500	1,000
JWS ↑	0.990	0.995	0.981	0.989	0.991

(b) Normalization ablation

	MNIST			**CIFAR-10**		
Method	**Full**	**Raw**	**No $\sqrt{3}$**	**Full**	**Raw**	**No $\sqrt{3}$**
Random init.	0.15	0.38	0.00	0.31	0.51	0.00
Perturbation	0.54	0.74	0.20	0.63	0.77	0.36
MLP	0.23	0.41	0.00	0.25	0.48	0.00
DiT	0.54	0.72	0.20	0.62	0.73	0.34
P-diff	0.25	0.42	0.00	0.29	0.50	0.00
DWF	0.27	0.42	0.00	0.37	0.54	0.00
DWF†	0.55	0.72	0.22	0.59	0.71	0.29
SANE Gauss†	0.29	0.46	0.00	0.39	0.55	0.00
SANE KDE†	0.59	0.77	0.28	0.64	0.78	0.37
GMN (ours)	**0.91**	**0.95**	**0.84**	**0.92**	**0.95**	**0.86**

Table 4: Classifier architectures and training settings. All affine layers include biases.

Setting	**MNIST MLP**	**CIFAR-10 CNN**
Collection size	10K	50K
Architecture	MLP $784 \to 32 \to 32 \to 10$, ReLU	Three 3×3 convolutions, channels $(16, 32, 48)$, padding 1, ReLU
Pooling and output	Linear classifier	2×2 max-pooling after the first two convolutions; global average pooling; linear classifier
Parameters	26,506	19,450
Optimizer	Adam	SGD; Nesterov momentum 0.9; weight decay 5×10^{-4}
Learning rate	10^{-3}, constant	0.1, cosine decay to zero
Batch size / epochs	64 / 5	128 / 60
Loss	Cross-entropy	Cross-entropy; label smoothing 0.1
Input preprocessing	Pixels in $[0, 1]$	Pixels in $[0, 1]$; channel normalization
Channel mean	–	$(0.4914, 0.4822, 0.4465)$
Channel std.	–	$(0.2470, 0.2435, 0.2616)$
Training augmentation	None	Random horizontal flip; 32×32 crop after four-pixel reflection padding

C Unconditional Generation

C.1 Data and Training Collections

We construct collections of independently trained classifiers using distinct random seeds and retain their final checkpoints. Weights and biases use PyTorch's default uniform initialization (Paszke et al., 2019). Table 4 summarizes the architectures and training recipes. Our CNNs use neither batch nor group normalization. The graph representation can nonetheless encode the affine parameters (γ, β) of normalization layers analogously to bias terms (Lim et al., 2024).

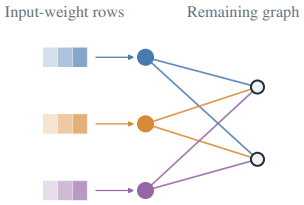

Figure 8: **Input-layer folding.** First-layer weights and biases are encoded as first-hidden-layer node features. Hidden-unit permutations reorder these features while leaving input coordinates fixed; shared encoders and decoders therefore preserve permutation equivariance.



Figure 9: **CNN kernel edge representation.** One edge carries a vectorized $k \times k$ kernel.

C.2 GMN Configuration

Both GMNs use the flow-matching objective in Equation 11, with independent standard Gaussian noise and $t \sim \mathcal{U}[0, 1]$. For these fixed-architecture experiments, we use input-layer folding on MNIST and vector-valued kernel edges on CIFAR-10, as described below. Neither adaptation is used in domain-conditioned generation (Section 5.2.2) or heterogeneous-architecture generation (Section 5.2.1).

MNIST MLP graph representation. Starting from the parameter-graph construction of Lim et al. (2024), we fold the first affine layer into node features to reduce message-passing cost (Figure 8). For each first-hidden-layer neuron, a shared linear encoder maps its 784 incoming weights and bias to a 128-dimensional feature. A shared linear decoder maps its final node feature to the corresponding 785 parameter velocities. The remaining 1,386 parameters are represented by scalar-valued edges on a graph with 74 neuron nodes and two bias nodes. The ODE state retains all 26,506 parameters.

CIFAR-10 CNN graph representation. We build on the parameter-graph construction of Lim et al. (2024), representing convolutional channels as nodes. Following Kofinas et al. (2024), each edge between an input and output channel carries the nine coefficients of their 3×3 kernel in a fixed spatial order (Figure 9). Shared encoders map these coefficients to edge features, and shared readouts predict the corresponding nine parameter velocities. Fully connected weights and biases are represented by scalar-valued edges. Hidden-channel permutations reorder nodes and their incident edges while leaving the spatial ordering within each kernel unchanged.

Training and sampling settings. Table 5 lists the GMN configurations. All sampling uses the exponential moving average (EMA) of the learned velocity field.

C.3 Parameter-Group Normalization

We normalize each layer's weights and biases separately. For each parameter group r, we compute a scalar mean μ_r and standard deviation σ_r by pooling all entries across the training collection, and apply

$$\widetilde{\boldsymbol{w}}_r = \frac{\boldsymbol{w}_r - \mu_r \mathbf{1}}{\sigma_r}.$$

These statistics are permutation-invariant and remain fixed during flow training and sampling. Generated weights are transformed back to their original scale before evaluation.

Table 5: GMN configurations for unconditional generation. Refinement uses corruption time t^* and k cycles.

Setting	MNIST	CIFAR-10
Architecture		
Message-passing blocks	8	16
Node / edge / time dimensions	128 / 128 / 128	192 / 384 / 128
Positional encodings	Orbit-shared	Orbit-shared
Weight normalization	Parameter-group	Parameter-group
Message aggregation	Sum	Sum
Global aggregation	Mean	Mean
Parameters (M)	8	77
Optimization		
Optimizer	AdamW	AdamW
Learning rate	10^{-3}	5×10^{-4}
Learning-rate schedule	Constant	Cosine decay
Warmup updates	0	2K
Weight decay	0	0
Global batch size	256	256
GPU model	NVIDIA RTX A6000	NVIDIA L40
Number of GPUs	4	8
Memory per GPU	48 GB	48 GB
Training time (wall-clock)	12 h	60 h
Training updates	200K	300K
Gradient clipping	Norm 1.0	Norm 1.0
EMA decay	0.999	0.999
Sampling		
Guidance scale (Ho & Salimans, 2022)	None	None
Generated networks	100	100
ODE solver	`dopri5`	`dopri5`
Refinement (t^*, k)	$(0.75, 10)$	$(0.3, 10)$
Wall-clock time per network		
Training from scratch	4 s	86 s
Generation	0.221 s	3.6 s
Generation + refinement	0.905 s	25 s
Both generation times are amortized over batches of 100 networks.		

C.4 Sampling and Refinement

Sampling integrates the EMA velocity field from Gaussian noise at $t = 0$ to $t = 1$. Each refinement cycle (Meng et al., 2022) draws fresh $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, corrupts the current normalized weights as

$$\widetilde{\boldsymbol{w}}_{t^*} = t^* \boldsymbol{w} + (1 - t^*)\boldsymbol{\epsilon},$$

and reintegrates from t^* to 1 using the same field. Refinement requires only repeated noise corruption and integration through the trained flow, with no access to task data or training networks and no additional training. We use ten cycles, with $t^* = 0.75$ for MNIST and $t^* = 0.3$ for CIFAR-10. CIFAR-10 refinement uses absolute and relative solver tolerances of 10^{-5}.

Table 6 isolates the effect of refinement. Even before refinement, our flow achieves higher JWS, by a large margin, than all evaluated baselines on both datasets (Table 1). Refinement further improves JWS and increases mean accuracy by 1.82 percentage points on MNIST and 3.23 points on CIFAR-10. The refined results are reported in Table 1.

C.5 Evaluation

All generated networks are evaluated on the standard test sets of MNIST and CIFAR-10, using the metrics and reference conventions in Appendix B. All WCS evaluations use five Git Re-Basin iterations per network pair, including training-reference and generated-network comparisons. This

Table 6: GMN refinement ablation. Each setting contains 100 generated networks; MNIST uses the same networks before and after refinement. Accuracy, Max IoU, and WCS report mean$_{\pm\text{std}}$ across networks; JWS uses repeated reference subsampling.

Sampling	Acc. (%)	Max IoU	WCS	JWS
	MNIST	$(t^*, k) = (0.75, 10)$		
Direct	$92.58_{\pm 0.89}$	$0.383_{\pm 0.029}$	$0.433_{\pm 0.007}$	0.776
Refined	$94.40_{\pm 0.41}$	$0.469_{\pm 0.022}$	$0.553_{\pm 0.008}$	0.906
	CIFAR-10	$(t^*, k) = (0.3, 10)$		
Direct	$72.11_{\pm 1.54}$	$0.496_{\pm 0.019}$	$0.601_{\pm 0.016}$	0.879
Refined	$75.34_{\pm 0.80}$	$0.533_{\pm 0.012}$	$0.630_{\pm 0.011}$	0.918

Table 7: Leave-one-out maximum similarity among generated networks, reported as mean$_{\pm\text{std}}$.

Dataset	Max IoU	WCS
MNIST	$0.418_{\pm 0.016}$	$0.529_{\pm 0.010}$
CIFAR-10	$0.506_{\pm 0.009}$	$0.610_{\pm 0.014}$

budget keeps full-collection evaluation tractable while screening for near-copies of training networks.

C.6 Diversity of Generated Networks

To check for mode collapse, we measure similarity among 100 generated networks per dataset. For each network, we compute its maximum IoU and WCS over the remaining 99 networks, excluding self-comparisons. Table 7 shows that these maximum similarities remain well below one, providing evidence against collapse to repeated or near-duplicate networks.

C.7 Baseline Configurations and Sweeps

Our baselines cover five families of weight-space generators. MLP and DiT provide non-equivariant architecture baselines, DWF performs flow matching directly on weights, and SANE and P-diff generate in latent space using fitted priors and learned diffusion, respectively. P-diff also represents the trajectory-based approach to a priori alignment, while SANE and DWF cover post-hoc alignment and permutation augmentation.

All baselines use the same independently trained MNIST and CIFAR-10 collections and evaluation protocol. Post-hoc alignment is applied only to designated baseline configurations; our method always uses unaligned collections except in the alignment ablation (Table 2). Before running the sweeps, we reproduce the baselines in their original settings.

Across these five families, we evaluate 110 configurations: 81 DWF, 18 SANE, 7 DiT, and one MLP and P-diff configuration per dataset. The searches vary model capacity, learning rate, batch size, schedule, EMA decay, alignment, and generation prior where applicable. Configuration counts include alternative sampling priors: SANE evaluates each of nine trained autoencoders with two priors. Table 8 summarizes the search coverage; Table 10 reports the outcomes and identifies the configurations selected for Table 1. The sweeps include baseline models with parameter counts below, comparable to, and above those of our GMN. We also evaluate random initialization and perturbed training networks as controls.

Random initialization and perturbations. As controls, we sample random weights and perturb training networks (Table 9). We select the perturbation level with the highest JWS: $\sigma = 0.25$ for MNIST and $\sigma = 0.10$ for CIFAR-10.

MLP. We use an MLP velocity network on vectorized, unaligned weights, with AdaLN time conditioning and no built-in permutation equivariance. The MNIST and CIFAR-10 models use hidden widths $352/1184$, $8/10$ blocks, and time-embedding dimensions $128/256$, respectively. Their pa-

Table 8: Baseline search coverage. Counts denote evaluated configurations, including generation-prior choices. The MNIST DWF search is a full Cartesian grid; the exact DiT and CIFAR-10 DWF combinations appear in Table 10.

Baseline	MNIST	CIFAR-10	Search coverage
MLP	1	1	Dataset-specific fixed architectures with 21.4M and 80.4M parameters, respectively.
DiT	5	2	MNIST: widths $\{256, 368\}$, learning rates $\{10^{-3}, 5 \times 10^{-4}, 3 \times 10^{-4}\}$, constant/cosine schedules, and EMA decays $\{0.999, 0.9999\}$ in five tested combinations. CIFAR-10: two width-864 configurations.
DWF	72	9	MNIST: full grid of 3 batch sizes, 3 learning rates, 4 widths, and 2 alignment settings. CIFAR-10: 9 combinations of 3 learning rates, 2 batch sizes, and 2 alignment settings.
SANE	14	4	Seven MNIST widths (2.0–202.7M parameters) and two CIFAR-10 widths (79.7–202.7M), each evaluated with KDE and Gaussian priors.
P-diff	1	1	Dataset-specific VAE/DDPM configurations, with 23.1M and 42.3M total parameters, respectively.
Total	**93**	**17**	**110** evaluated configurations.

Table 9: Random initialization and perturbation controls. Entries report mean $\pm$ std across networks, or reference subsamples for JWS. Bold marks the highest JWS per dataset.

Control	Acc. (%)	Max IoU	WCS	JWS
MNIST				
Random init.	$10.11_{\pm 2.39}$	$0.052_{\pm 0.002}$	$0.138_{\pm 0.003}$	$0.1457_{\pm 0.0010}$
$\sigma = 0.05$	$95.81_{\pm 0.25}$	$0.855_{\pm 0.029}$	$0.999_{\pm 0.000}$	$0.4482_{\pm 0.0011}$
$\sigma = 0.10$	$95.52_{\pm 0.35}$	$0.717_{\pm 0.047}$	$0.995_{\pm 0.000}$	$0.5155_{\pm 0.0010}$
$\sigma = 0.25$	$92.17_{\pm 2.15}$	$0.403_{\pm 0.073}$	$0.971_{\pm 0.001}$	$\mathbf{0.5366}_{\pm 0.0010}$
$\sigma = 0.50$	$76.24_{\pm 7.42}$	$0.166_{\pm 0.044}$	$0.896_{\pm 0.003}$	$0.4471_{\pm 0.0012}$
CIFAR-10				
Random init.	$10.09_{\pm 1.71}$	$0.254_{\pm 0.012}$	$0.255_{\pm 0.006}$	$0.3071_{\pm 0.0008}$
$\sigma = 0.05$	$78.29_{\pm 0.54}$	$0.767_{\pm 0.031}$	$0.999_{\pm 0.000}$	$0.5998_{\pm 0.0012}$
$\sigma = 0.10$	$74.84_{\pm 1.72}$	$0.610_{\pm 0.047}$	$0.995_{\pm 0.000}$	$\mathbf{0.6296}_{\pm 0.0012}$
$\sigma = 0.25$	$53.66_{\pm 6.59}$	$0.362_{\pm 0.029}$	$0.971_{\pm 0.001}$	$0.5413_{\pm 0.0008}$
$\sigma = 0.50$	$24.10_{\pm 5.03}$	$0.278_{\pm 0.015}$	$0.896_{\pm 0.002}$	$0.4204_{\pm 0.0005}$

rameter counts are 21,434,730 and 80,356,378. Both use AdamW with learning rate 2×10^{-4}, batch size 64, EMA decay 0.999, and parameter-group normalization. Each configuration generates 100 networks.

DiT. The DiT velocity network divides vectorized weights into fixed-size chunks and uses learned positional embeddings and AdaLN time conditioning. It operates on unaligned weights without built-in permutation equivariance. All configurations use 8 transformer blocks and batch size 256. Widths 256, 368, and 864 give 8,223,008, 15,762,848, and 77,913,152 parameters, respectively. Training budgets are 200K updates for MNIST and 300K for CIFAR-10; checkpoints are selected according to generation quality. We use the official implementation by Peebles & Xie (2023). The configuration that achieved the highest mean JWS determines the entry in Table 1. We note that the best run for DiT on MNIST classifiers has two times the amount of parameters as our GMN (Table 5), and still didn't close the gap. All runs use the AdamW optimizer without weight decay (in line with Peebles & Xie (2023)). Each configuration generates 100 networks.

DWF. We use the authors' implementation (Gupta et al., 2026) directly on vectorized weights without PCA, with and without Git Re-Basin alignment. The MNIST grid uses $b \in \{32, 64, 256\}$, $\eta \in \{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$, and widths $h \in \{200, 300, 400, 558\}$ in both alignment settings, with

200K updates per run. The widths $200, 300, 400, 558$ give 10,687,494, 16,045,844, 21,424,194, and 29,962,751 parameters, respectively. CIFAR-10 uses width 1954 (79,992,165 parameters) and 300K updates per run. At batch size 256, all three learning rates are tested in both alignment settings; at batch size 64, $\eta = 5 \times 10^{-4}$ is tested in both settings and $\eta = 10^{-4}$ only with alignment. Remaining settings follow the authors' implementation. At each configuration we evaluate 100 generated networks.

SANE. We use the authors' implementation (Schürholt et al., 2024) with permutation augmentation on Git Re-Basin-aligned collections. MNIST models use 6 transformer layers and widths $\{104, 160, 224, 232, 320, 328, 1024\}$; CIFAR-10 models use 8 layers and widths $\{640, 1024\}$. Each autoencoder is evaluated with KDE and Gaussian priors, generating 100 networks per configuration. For each dataset and prior, the highest mean JWS determines the entry in Table 1. Remaining settings follow the authors' implementation.

P-diff. We use the authors' implementation (Wang et al., 2024) on unaligned collections. The MNIST model has 23.1M parameters (15.4M VAE and 7.7M DDPM), while the CIFAR-10 model has 42.3M (32.7M VAE and 9.6M DDPM). MNIST follows the authors' recipe with 2K VAE and 8K DDPM updates at batch size 50. CIFAR-10 uses 20K VAE and 300K DDPM updates at batch size 256, matching the DDPM budget and batch size to our GMN. Each configuration generates 100 networks.

Table 10: Consolidated baseline configurations and sweep results. h denotes hidden or transformer width, η learning rate, ρ EMA decay. b batch size, L the number of blocks, and d_t the time-embedding dimension. Parameter counts are approximate and given in millions. Entries use $x_{\pm s} \equiv x \pm s$; accuracy, Max IoU, and WCS summarize generated networks. JWS is reported as a mean with subsampling standard deviation. Bold marks selected configurations reported in Table 1. The 36 unaligned MNIST DWF runs are summarized by per-metric extrema, which need not correspond to the same configuration.

Configuration	Params (M)	Acc. (%)	Max IoU	WCS	JWS
MLP: MNIST					
$h = 352,\ L = 8$ $d_t = 128$	21.4	$9.78_{\pm 3.74}$	$0.0517_{\pm 0.0020}$	$0.3554_{\pm 0.0086}$	$\mathbf{0.2335_{\pm 0.0010}}$
MLP: CIFAR-10					
$h = 1184,\ L = 10$ $d_t = 256$	80.4	$9.98_{\pm 1.55}$	$0.2558_{\pm 0.0120}$	$0.1464_{\pm 0.0025}$	$\mathbf{0.2520_{\pm 0.0009}}$
DiT: MNIST					
$h = 256,\ \eta = 10^{-3}$ Cosine; $\rho = 0.999$	8.2	$9.95_{\pm 5.40}$	$0.0517_{\pm 0.026}$	$0.5345_{\pm 0.009}$	$0.2625_{\pm 0.0010}$
$h = 256,\ \eta = 5 \times 10^{-4}$ Cosine; $\rho = 0.999$	8.2	$9.10_{\pm 4.73}$	$0.0513_{\pm 0.024}$	$0.5324_{\pm 0.009}$	$0.2587_{\pm 0.0010}$
$h = 368,\ \eta = 10^{-3}$ Cosine; $\rho = 0.999$	15.8	$71.97_{\pm 6.45}$	$0.1381_{\pm 0.032}$	$0.5553_{\pm 0.010}$	$\mathbf{0.5401_{\pm 0.0013}}$
$h = 368,\ \eta = 5 \times 10^{-4}$ Cosine; $\rho = 0.999$	15.8	$62.78_{\pm 8.93}$	$0.1086_{\pm 0.024}$	$0.5509_{\pm 0.009}$	$0.4910_{\pm 0.0013}$
$h = 368,\ \eta = 3 \times 10^{-4}$ Const.; $\rho = 0.9999$	15.8	$66.06_{\pm 7.49}$	$0.1179_{\pm 0.024}$	$0.5515_{\pm 0.009}$	$0.5081_{\pm 0.0013}$
DiT: CIFAR-10					
$h = 864,\ \eta = 2 \times 10^{-4}$ Const.; $\rho = 0.999$	77.9	$32.47_{\pm 5.98}$	$0.2952_{\pm 0.019}$	$0.5377_{\pm 0.015}$	$0.5400_{\pm 0.0006}$
$h = 864,\ \eta = 5 \times 10^{-4}$ Cosine.; $\rho = 0.999$	77.9	$43.67_{\pm 5.52}$	$0.326_{\pm 0.025}$	$0.553_{\pm 0.015}$	$\mathbf{0.620_{\pm 0.0006}}$
DWF: MNIST, aligned, $b = 32$					
$h = 200,\ \eta = 10^{-3}$	10.7	$72.78_{\pm 6.67}$	$0.1452_{\pm 0.033}$	$0.6270_{\pm 0.028}$	$0.5421_{\pm 0.0013}$
$h = 300,\ \eta = 10^{-3}$	16.0	$71.82_{\pm 8.29}$	$0.1419_{\pm 0.035}$	$0.5936_{\pm 0.025}$	$0.5403_{\pm 0.0013}$

Continued on next page

Table 10 (continued)

Configuration	Params (M)	Acc. (%)	Max IoU	WCS	JWS
$h = 400,\ \eta = 10^{-3}$	21.4	$72.11_{\pm 7.17}$	$0.1416_{\pm 0.032}$	$0.5848_{\pm 0.012}$	$0.5425_{\pm 0.0013}$
$h = 558,\ \eta = 10^{-3}$	30.0	$65.18_{\pm 12.32}$	$0.1190_{\pm 0.031}$	$0.5563_{\pm 0.044}$	$0.5020_{\pm 0.0013}$
$h = 200,\ \eta = 5 \times 10^{-4}$	10.7	$70.43_{\pm 10.66}$	$0.1377_{\pm 0.034}$	$0.5955_{\pm 0.029}$	$0.5315_{\pm 0.0013}$
$h = 300,\ \eta = 5 \times 10^{-4}$	16.0	$72.85_{\pm 6.54}$	$0.1454_{\pm 0.034}$	$0.5952_{\pm 0.015}$	$\mathbf{0.5470_{\pm 0.0013}}$
$h = 400,\ \eta = 5 \times 10^{-4}$	21.4	$69.71_{\pm 7.38}$	$0.1317_{\pm 0.030}$	$0.5724_{\pm 0.015}$	$0.5288_{\pm 0.0013}$
$h = 558,\ \eta = 5 \times 10^{-4}$	30.0	$54.06_{\pm 12.37}$	$0.0911_{\pm 0.019}$	$0.5217_{\pm 0.042}$	$0.4476_{\pm 0.0012}$
$h = 200,\ \eta = 10^{-4}$	10.7	$71.43_{\pm 5.93}$	$0.1365_{\pm 0.026}$	$0.5947_{\pm 0.015}$	$0.5362_{\pm 0.0013}$
$h = 300,\ \eta = 10^{-4}$	16.0	$72.86_{\pm 6.72}$	$0.1438_{\pm 0.033}$	$0.5861_{\pm 0.011}$	$0.5461_{\pm 0.0013}$
$h = 400,\ \eta = 10^{-4}$	21.4	$70.38_{\pm 6.68}$	$0.1338_{\pm 0.031}$	$0.5728_{\pm 0.010}$	$0.5325_{\pm 0.0013}$
$h = 558,\ \eta = 10^{-4}$	30.0	$64.31_{\pm 6.83}$	$0.1109_{\pm 0.021}$	$0.5542_{\pm 0.011}$	$0.4980_{\pm 0.0013}$
DWF: MNIST, aligned, $b = 64$					
$h = 200,\ \eta = 10^{-3}$	10.7	$51.78_{\pm 16.84}$	$0.0927_{\pm 0.030}$	$0.5239_{\pm 0.053}$	$0.4375_{\pm 0.0012}$
$h = 300,\ \eta = 10^{-3}$	16.0	$46.62_{\pm 14.84}$	$0.0811_{\pm 0.020}$	$0.5152_{\pm 0.029}$	$0.4153_{\pm 0.0012}$
$h = 400,\ \eta = 10^{-3}$	21.4	$23.58_{\pm 9.19}$	$0.0584_{\pm 0.007}$	$0.4543_{\pm 0.041}$	$0.3128_{\pm 0.0010}$
$h = 558,\ \eta = 10^{-3}$	30.0	$22.83_{\pm 7.55}$	$0.0572_{\pm 0.005}$	$0.4645_{\pm 0.017}$	$0.3120_{\pm 0.0010}$
$h = 200,\ \eta = 5 \times 10^{-4}$	10.7	$65.01_{\pm 9.04}$	$0.1156_{\pm 0.032}$	$0.5758_{\pm 0.020}$	$0.5019_{\pm 0.0013}$
$h = 300,\ \eta = 5 \times 10^{-4}$	16.0	$61.60_{\pm 12.07}$	$0.1099_{\pm 0.033}$	$0.5564_{\pm 0.022}$	$0.4866_{\pm 0.0012}$
$h = 400,\ \eta = 5 \times 10^{-4}$	21.4	$56.42_{\pm 13.54}$	$0.0977_{\pm 0.027}$	$0.5402_{\pm 0.023}$	$0.4608_{\pm 0.0012}$
$h = 558,\ \eta = 5 \times 10^{-4}$	30.0	$44.30_{\pm 13.56}$	$0.0776_{\pm 0.019}$	$0.5130_{\pm 0.022}$	$0.4062_{\pm 0.0012}$
$h = 200,\ \eta = 10^{-4}$	10.7	$72.35_{\pm 7.15}$	$0.1424_{\pm 0.032}$	$0.5891_{\pm 0.016}$	$0.5434_{\pm 0.0013}$
$h = 300,\ \eta = 10^{-4}$	16.0	$68.36_{\pm 7.38}$	$0.1243_{\pm 0.029}$	$0.5751_{\pm 0.014}$	$0.5188_{\pm 0.0013}$
$h = 400,\ \eta = 10^{-4}$	21.4	$67.45_{\pm 8.57}$	$0.1227_{\pm 0.029}$	$0.5612_{\pm 0.012}$	$0.5149_{\pm 0.0013}$
$h = 558,\ \eta = 10^{-4}$	30.0	$60.68_{\pm 9.04}$	$0.1035_{\pm 0.023}$	$0.5447_{\pm 0.010}$	$0.4809_{\pm 0.0012}$
DWF: MNIST, aligned, $b = 256$					
$h = 200,\ \eta = 10^{-3}$	10.7	$39.64_{\pm 13.44}$	$0.0716_{\pm 0.018}$	$0.5057_{\pm 0.028}$	$0.3853_{\pm 0.0011}$
$h = 300,\ \eta = 10^{-3}$	16.0	$22.40_{\pm 8.37}$	$0.0574_{\pm 0.006}$	$0.4737_{\pm 0.022}$	$0.3113_{\pm 0.0010}$
$h = 400,\ \eta = 10^{-3}$	21.4	$17.78_{\pm 6.42}$	$0.0550_{\pm 0.004}$	$0.4524_{\pm 0.026}$	$0.2889_{\pm 0.0010}$
$h = 558,\ \eta = 10^{-3}$	30.0	$12.31_{\pm 4.24}$	$0.0525_{\pm 0.002}$	$0.4438_{\pm 0.013}$	$0.2643_{\pm 0.0010}$
$h = 200,\ \eta = 5 \times 10^{-4}$	10.7	$56.41_{\pm 12.71}$	$0.0967_{\pm 0.026}$	$0.5400_{\pm 0.026}$	$0.4605_{\pm 0.0012}$
$h = 300,\ \eta = 5 \times 10^{-4}$	16.0	$60.77_{\pm 10.48}$	$0.1046_{\pm 0.025}$	$0.5466_{\pm 0.029}$	$0.4808_{\pm 0.0012}$
$h = 400,\ \eta = 5 \times 10^{-4}$	21.4	$42.34_{\pm 14.58}$	$0.0757_{\pm 0.018}$	$0.5082_{\pm 0.029}$	$0.3969_{\pm 0.0011}$
$h = 558,\ \eta = 5 \times 10^{-4}$	30.0	$28.72_{\pm 8.61}$	$0.0614_{\pm 0.007}$	$0.4827_{\pm 0.018}$	$0.3395_{\pm 0.0011}$
$h = 200,\ \eta = 10^{-4}$	10.7	$68.94_{\pm 8.43}$	$0.1267_{\pm 0.031}$	$0.5826_{\pm 0.016}$	$0.5212_{\pm 0.0013}$
$h = 300,\ \eta = 10^{-4}$	16.0	$62.69_{\pm 10.38}$	$0.1107_{\pm 0.030}$	$0.5578_{\pm 0.015}$	$0.4915_{\pm 0.0013}$
$h = 400,\ \eta = 10^{-4}$	21.4	$60.15_{\pm 10.33}$	$0.1029_{\pm 0.025}$	$0.5481_{\pm 0.014}$	$0.4782_{\pm 0.0012}$
$h = 558,\ \eta = 10^{-4}$	30.0	$53.24_{\pm 10.10}$	$0.0877_{\pm 0.017}$	$0.5272_{\pm 0.014}$	$0.4456_{\pm 0.0012}$
DWF: MNIST, unaligned (36 **configurations; per-metric extrema)**					
Minimum	—	9.28	0.051	0.52	$0.2599_{\pm 0.0010}$
Maximum	—	11.05	0.052	0.55	$0.2670_{\pm 0.0010}$
DWF: CIFAR-10, aligned, $h = 1954$					
$b = 256,\ \eta = 5 \times 10^{-4}$	80.0	$27.84_{\pm 9.40}$	$0.2891_{\pm 0.021}$	$0.6002_{\pm 0.030}$	$0.5124_{\pm 0.0005}$
$b = 256,\ \eta = 10^{-3}$	80.0	$32.61_{\pm 7.93}$	$0.2952_{\pm 0.022}$	$0.6199_{\pm 0.022}$	$0.5439_{\pm 0.0005}$
$b = 256,\ \eta = 10^{-4}$	80.0	$38.68_{\pm 8.16}$	$0.3125_{\pm 0.024}$	$0.6467_{\pm 0.023}$	$0.5853_{\pm 0.0004}$
$b = 64,\ \eta = 5 \times 10^{-4}$	80.0	$34.56_{\pm 5.62}$	$0.2971_{\pm 0.019}$	$0.6422_{\pm 0.017}$	$0.5565_{\pm 0.0004}$
$b = 64,\ \eta = 10^{-4}$	80.0	$39.60_{\pm 6.62}$	$0.3151_{\pm 0.024}$	$0.6713_{\pm 0.019}$	$\mathbf{0.5905_{\pm 0.0004}}$
DWF: CIFAR-10, unaligned, $h = 1954$					
$b = 256,\ \eta = 5 \times 10^{-4}$	80.0	$10.00_{\pm 0.10}$	$0.2487_{\pm 0.001}$	$0.4320_{\pm 0.009}$	$\mathbf{0.3679_{\pm 0.0006}}$
$b = 256,\ \eta = 10^{-3}$	80.0	$9.97_{\pm 0.25}$	$0.2487_{\pm 0.001}$	$0.4299_{\pm 0.007}$	$0.3671_{\pm 0.0006}$
$b = 256,\ \eta = 10^{-4}$	80.0	$10.05_{\pm 0.46}$	$0.2488_{\pm 0.001}$	$0.4251_{\pm 0.007}$	$0.3664_{\pm 0.0006}$
$b = 64,\ \eta = 5 \times 10^{-4}$	80.0	$10.00_{\pm 0.02}$	$0.2487_{\pm 0.001}$	$0.4252_{\pm 0.007}$	$0.3661_{\pm 0.0006}$
SANE: MNIST, aligned					
$h = 104$, KDE	2.0	$93.86_{\pm 1.57}$	$0.4701_{\pm 0.066}$	$0.9412_{\pm 0.004}$	$\mathbf{0.5868_{\pm 0.0009}}$

Continued on next page

Table 10 (continued)

Configuration	Params (M)	Acc. (%)	Max IoU	WCS	JWS
$h = 104$, Gauss.	2.0	$11.63_{\pm 4.35}$	$0.0523_{\pm 0.002}$	$0.5344_{\pm 0.017}$	$0.2702_{\pm 0.0010}$
$h = 160$, KDE	4.3	$94.99_{\pm 0.55}$	$0.5638_{\pm 0.054}$	$0.9548_{\pm 0.003}$	$0.5847_{\pm 0.0009}$
$h = 160$, Gauss.	4.3	$16.88_{\pm 5.45}$	$0.0544_{\pm 0.003}$	$0.5310_{\pm 0.018}$	$\mathbf{0.2931_{\pm 0.0010}}$
$h = 224$, KDE	7.9	$95.15_{\pm 0.54}$	$0.5874_{\pm 0.057}$	$0.9596_{\pm 0.002}$	$0.5785_{\pm 0.0009}$
$h = 224$, Gauss.	7.9	$10.77_{\pm 4.27}$	$0.0520_{\pm 0.002}$	$0.5789_{\pm 0.016}$	$0.2664_{\pm 0.0010}$
$h = 232$, KDE	8.5	$95.17_{\pm 0.62}$	$0.5887_{\pm 0.057}$	$0.9607_{\pm 0.002}$	$0.5772_{\pm 0.0009}$
$h = 232$, Gauss.	8.5	$10.75_{\pm 3.78}$	$0.0521_{\pm 0.002}$	$0.5630_{\pm 0.016}$	$0.2668_{\pm 0.0010}$
$h = 320$, KDE	15.7	$95.58_{\pm 0.25}$	$0.6733_{\pm 0.042}$	$0.9693_{\pm 0.001}$	$0.5548_{\pm 0.0009}$
$h = 320$, Gauss.	15.7	$11.24_{\pm 3.59}$	$0.0526_{\pm 0.002}$	$0.5902_{\pm 0.016}$	$0.2685_{\pm 0.0010}$
$h = 328$, KDE	16.4	$95.44_{\pm 0.40}$	$0.6543_{\pm 0.055}$	$0.9705_{\pm 0.001}$	$0.5574_{\pm 0.0009}$
$h = 328$, Gauss.	16.4	$11.55_{\pm 3.49}$	$0.0526_{\pm 0.002}$	$0.5909_{\pm 0.014}$	$0.2698_{\pm 0.0010}$
$h = 1024$, KDE	202.7	$95.72_{\pm 0.27}$	$0.7637_{\pm 0.041}$	$0.9754_{\pm 0.001}$	$0.5159_{\pm 0.0010}$
$h = 1024$, Gauss.	202.7	$14.30_{\pm 3.87}$	$0.0534_{\pm 0.002}$	$0.3445_{\pm 0.012}$	$0.2497_{\pm 0.0010}$
SANE: CIFAR-10, aligned					
$h = 640$, KDE	79.7	$72.91_{\pm 2.52}$	$0.5546_{\pm 0.054}$	$0.9821_{\pm 0.002}$	$\mathbf{0.6356_{\pm 0.0011}}$
$h = 640$, Gauss.	79.7	$10.92_{\pm 1.88}$	$0.2554_{\pm 0.009}$	$0.4819_{\pm 0.014}$	$\mathbf{0.3875_{\pm 0.0006}}$
$h = 1024$, KDE	202.7	$76.00_{\pm 1.24}$	$0.6465_{\pm 0.047}$	$0.9897_{\pm 0.001}$	$0.6338_{\pm 0.0012}$
$h = 1024$, Gauss.	202.7	$11.16_{\pm 2.09}$	$0.2565_{\pm 0.010}$	$0.4673_{\pm 0.015}$	$0.3866_{\pm 0.0006}$
P-diff: MNIST					
VAE/DDPM: 2K/8K $b = 50$	23.1	$9.86_{\pm 0.63}$	$0.0517_{\pm 0.0004}$	$0.4038_{\pm 0.0091}$	$\mathbf{0.2463_{\pm 0.0010}}$
P-diff: CIFAR-10					
VAE/DDPM: 20K/300K $b = 256$	42.3	$10.01_{\pm 0.09}$	$0.2487_{\pm 0.0003}$	$0.2339_{\pm 0.0036}$	$\mathbf{0.2944_{\pm 0.0008}}$

D Sample and Model Complexity

Data and collections. We independently train 50K MLPs with architecture $64 \to 16 \to 8 \to 10$, ReLU hidden activations, and 1,266 parameters on `mfeat-karhunen` (Bischl et al., 2021). Networks differ in training seed and share one stratified $80/20$ split (1,600 training and 400 evaluation examples). Imputation and standardization are fitted on the training split and frozen across the collection. Generators use nested checkpoint subsets with identical indices across methods. Aligned DWF uses the same indices after Git Re-Basin alignment (Ainsworth et al., 2022) to a common reference.

Generators. We compare a 2M GMN with DiT and DWF at approximately 2M and 10M parameters, using the same GMN in both comparisons. Both DWF capacities use aligned and unaligned collections, yielding seven configurations per collection size and 70 in total. We use the authors' DWF implementation (Gupta et al., 2026). Table 11 gives all reported hyperparameters. We use no early stopping or validation-based checkpoint selection.

Table 11: Hyperparameters for the scaling experiment. Width denotes the GMN feature dimension, DiT hidden dimension, or DWF hidden width. Orbit-shared encodings use hidden-unit permutation orbits.

Setting	GMN	DiT	DiT-L	DWF	DWF-L
Shared settings					
Collection sizes N	100, 200, 400, 800, 1,500, 3,000, 6,000, 12,000, 25,000, 50,000				
Training updates	75K per configuration				
Global batch size	256				
Learning rate	5×10^{-4}				
Learning-rate schedule	Cosine decay to 10^{-6}				
Gradient clipping	Global norm 1.0				
Checkpoint	Final; EMA weights where specified below				
Generated networks	100 per configuration				
Integration steps	100				
Architecture and representation					
Parameters	1,958,017	2,019,872	10,731,424	2,009,538	10,557,490
Feature / hidden width	64	128	256	608	2,176
Blocks	8	8	12	–	–
Parameter representation	Graph	40 tokens of size 32		Vectorized	
Node / edge dimensions	64/64	–	–	–	–
Time dimension	64	64	64	128	128
Attention heads	–	8	8	–	–
Message aggregation	Sum	–	–	–	–
Positional encodings	Orbit-shared	Learned		–	–
Time conditioning	AdaLN	AdaLN		–	–
Weight normalization	Parameter-group			–	–
Collection alignment	None			With / without	
Matching iterations	–	–	–	100 (aligned)	
Dropout	–	–	–	0.1	0.1
Optimization and sampling					
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-5}	10^{-5}
EMA decay	0.999	0.999	0.999	None	None
ODE solver	Euler	Euler	Euler	RK4	RK4

Evaluation. All methods use the same held-out examples and the metrics in Appendix B. Max IoU and matched WCS use the entire 50K collection at every N, including networks outside the generator's training subset. JWS uses a fixed cloud of $1{,}000$ reference networks, excluding self-comparisons from their similarity statistics. The coordinate scales are fixed at approximately $(0.890, 0.674, 0.625)$ for every N. Table 12 reports all 70 configurations, complementing Figure 5.

Table 12: Scaling results for all seven configurations. Accuracy, Max IoU, and WCS report $x_{\pm s} \equiv x \pm s$ across 100 generated networks; JWS measures joint distributional agreement with the fixed reference cloud. Models have approximately 2M parameters, or 10M for the -L variants. † denotes Git Re-Basin alignment of the training collection. Bold marks the highest JWS at each N.

N	Model	Acc. (%)	Max IoU	WCS	JWS
100	GMN	$62.25_{\pm 11.81}$	$0.519_{\pm 0.067}$	$0.792_{\pm 0.051}$	$\mathbf{0.721}$
	DiT	$89.24_{\pm 2.47}$	$0.979_{\pm 0.025}$	$1.000_{\pm 0.000}$	0.564
	DiT-L	$86.24_{\pm 13.33}$	$0.974_{\pm 0.101}$	$0.988_{\pm 0.069}$	0.558
	DWF	$85.15_{\pm 17.45}$	$0.860_{\pm 0.117}$	$0.971_{\pm 0.120}$	0.605
	DWF†	$89.39_{\pm 2.33}$	$0.906_{\pm 0.068}$	$0.998_{\pm 0.006}$	0.600
	DWF-L	$88.69_{\pm 3.03}$	$0.897_{\pm 0.076}$	$0.998_{\pm 0.002}$	0.603
	DWF-L†	$88.48_{\pm 3.11}$	$0.873_{\pm 0.070}$	$0.997_{\pm 0.006}$	0.614
200	GMN	$68.39_{\pm 9.73}$	$0.543_{\pm 0.062}$	$0.692_{\pm 0.032}$	$\mathbf{0.806}$
	DiT	$72.68_{\pm 25.73}$	$0.745_{\pm 0.221}$	$0.916_{\pm 0.133}$	0.615
	DiT-L	$80.46_{\pm 24.13}$	$0.924_{\pm 0.179}$	$0.958_{\pm 0.117}$	0.560
	DWF	$88.28_{\pm 3.42}$	$0.895_{\pm 0.062}$	$0.998_{\pm 0.002}$	0.605
	DWF†	$88.77_{\pm 2.44}$	$0.880_{\pm 0.066}$	$0.998_{\pm 0.002}$	0.611
	DWF-L	$88.70_{\pm 3.87}$	$0.854_{\pm 0.095}$	$0.995_{\pm 0.020}$	0.621
	DWF-L†	$88.55_{\pm 3.14}$	$0.874_{\pm 0.079}$	$0.997_{\pm 0.008}$	0.614
400	GMN	$76.82_{\pm 7.32}$	$0.577_{\pm 0.068}$	$0.663_{\pm 0.013}$	$\mathbf{0.873}$
	DiT	$12.79_{\pm 7.10}$	$0.425_{\pm 0.048}$	$0.589_{\pm 0.011}$	0.459
	DiT-L	$86.82_{\pm 11.83}$	$0.961_{\pm 0.111}$	$0.989_{\pm 0.060}$	0.566
	DWF	$88.27_{\pm 7.10}$	$0.859_{\pm 0.087}$	$0.994_{\pm 0.032}$	0.617
	DWF†	$87.21_{\pm 5.13}$	$0.806_{\pm 0.128}$	$0.983_{\pm 0.041}$	0.640
	DWF-L	$88.38_{\pm 3.27}$	$0.843_{\pm 0.086}$	$0.997_{\pm 0.004}$	0.624
	DWF-L†	$88.19_{\pm 3.53}$	$0.830_{\pm 0.103}$	$0.993_{\pm 0.014}$	0.630
800	GMN	$79.97_{\pm 5.09}$	$0.571_{\pm 0.064}$	$0.646_{\pm 0.012}$	$\mathbf{0.888}$
	DiT	$11.14_{\pm 6.97}$	$0.418_{\pm 0.039}$	$0.576_{\pm 0.012}$	0.446
	DiT-L	$75.15_{\pm 25.56}$	$0.819_{\pm 0.229}$	$0.927_{\pm 0.136}$	0.592
	DWF	$20.11_{\pm 6.95}$	$0.405_{\pm 0.044}$	$0.539_{\pm 0.039}$	0.489
	DWF†	$56.94_{\pm 7.23}$	$0.495_{\pm 0.050}$	$0.664_{\pm 0.057}$	0.733
	DWF-L	$85.14_{\pm 10.73}$	$0.757_{\pm 0.138}$	$0.979_{\pm 0.053}$	0.646
	DWF-L†	$85.68_{\pm 6.43}$	$0.743_{\pm 0.132}$	$0.968_{\pm 0.063}$	0.663
1,500	GMN	$83.16_{\pm 4.19}$	$0.580_{\pm 0.060}$	$0.635_{\pm 0.012}$	$\mathbf{0.908}$
	DiT	$9.95_{\pm 6.13}$	$0.427_{\pm 0.037}$	$0.564_{\pm 0.013}$	0.441
	DiT-L	$18.67_{\pm 14.96}$	$0.428_{\pm 0.073}$	$0.638_{\pm 0.073}$	0.485
	DWF	$13.88_{\pm 6.25}$	$0.418_{\pm 0.044}$	$0.522_{\pm 0.014}$	0.456
	DWF†	$50.81_{\pm 8.40}$	$0.469_{\pm 0.050}$	$0.603_{\pm 0.014}$	0.692
	DWF-L	$61.65_{\pm 26.06}$	$0.567_{\pm 0.168}$	$0.860_{\pm 0.142}$	0.628
	DWF-L†	$73.35_{\pm 7.98}$	$0.568_{\pm 0.073}$	$0.841_{\pm 0.073}$	0.746
3,000	GMN	$86.47_{\pm 3.02}$	$0.609_{\pm 0.059}$	$0.632_{\pm 0.012}$	$\mathbf{0.939}$
	DiT	$10.64_{\pm 6.38}$	$0.417_{\pm 0.039}$	$0.554_{\pm 0.013}$	0.441
	DiT-L	$13.57_{\pm 5.80}$	$0.419_{\pm 0.040}$	$0.603_{\pm 0.012}$	0.463
	DWF	$15.88_{\pm 6.74}$	$0.420_{\pm 0.044}$	$0.530_{\pm 0.015}$	0.469
	DWF†	$49.17_{\pm 8.72}$	$0.469_{\pm 0.054}$	$0.600_{\pm 0.018}$	0.683
	DWF-L	$12.09_{\pm 3.51}$	$0.446_{\pm 0.049}$	$0.509_{\pm 0.026}$	0.452
	DWF-L†	$80.40_{\pm 7.21}$	$0.583_{\pm 0.074}$	$0.916_{\pm 0.077}$	0.704

Continued on next page

Table 12 (continued)

N	Model	Acc. (%)	Max IoU	WCS	JWS
6,000	GMN	$87.04_{\pm 3.07}$	$0.621_{\pm 0.060}$	$0.625_{\pm 0.014}$	**0.949**
	DiT	$11.69_{\pm 6.26}$	$0.423_{\pm 0.041}$	$0.549_{\pm 0.012}$	0.449
	DiT-L	$19.67_{\pm 7.67}$	$0.415_{\pm 0.043}$	$0.589_{\pm 0.016}$	0.495
	DWF	$14.45_{\pm 6.40}$	$0.420_{\pm 0.041}$	$0.544_{\pm 0.015}$	0.463
	DWF†	$58.61_{\pm 8.61}$	$0.506_{\pm 0.059}$	$0.657_{\pm 0.018}$	0.750
	DWF-L	$19.35_{\pm 6.50}$	$0.410_{\pm 0.047}$	$0.520_{\pm 0.015}$	0.484
	DWF-L†	$48.86_{\pm 7.86}$	$0.465_{\pm 0.050}$	$0.594_{\pm 0.014}$	0.680
12,000	GMN	$87.13_{\pm 2.69}$	$0.617_{\pm 0.062}$	$0.624_{\pm 0.016}$	**0.946**
	DiT	$9.69_{\pm 5.94}$	$0.420_{\pm 0.033}$	$0.551_{\pm 0.013}$	0.436
	DiT-L	$72.27_{\pm 6.91}$	$0.549_{\pm 0.060}$	$0.627_{\pm 0.012}$	0.843
	DWF	$12.38_{\pm 5.92}$	$0.422_{\pm 0.043}$	$0.546_{\pm 0.014}$	0.452
	DWF†	$60.79_{\pm 7.71}$	$0.516_{\pm 0.057}$	$0.666_{\pm 0.017}$	0.765
	DWF-L	$15.57_{\pm 6.88}$	$0.415_{\pm 0.040}$	$0.519_{\pm 0.015}$	0.464
	DWF-L†	$49.36_{\pm 8.61}$	$0.457_{\pm 0.041}$	$0.598_{\pm 0.014}$	0.678
25,000	GMN	$87.60_{\pm 2.88}$	$0.630_{\pm 0.055}$	$0.626_{\pm 0.013}$	**0.958**
	DiT	$10.60_{\pm 5.59}$	$0.415_{\pm 0.041}$	$0.549_{\pm 0.011}$	0.440
	DiT-L	$76.46_{\pm 6.49}$	$0.545_{\pm 0.060}$	$0.620_{\pm 0.014}$	0.859
	DWF	$13.70_{\pm 7.10}$	$0.417_{\pm 0.041}$	$0.546_{\pm 0.016}$	0.458
	DWF†	$63.31_{\pm 6.75}$	$0.525_{\pm 0.065}$	$0.680_{\pm 0.020}$	0.780
	DWF-L	$14.06_{\pm 6.15}$	$0.420_{\pm 0.045}$	$0.522_{\pm 0.017}$	0.458
	DWF-L†	$56.93_{\pm 8.09}$	$0.495_{\pm 0.059}$	$0.628_{\pm 0.014}$	0.738
50,000	GMN	$87.22_{\pm 2.53}$	$0.627_{\pm 0.053}$	$0.625_{\pm 0.014}$	**0.955**
	DiT	$10.62_{\pm 6.15}$	$0.421_{\pm 0.038}$	$0.546_{\pm 0.013}$	0.441
	DiT-L	$82.41_{\pm 4.50}$	$0.564_{\pm 0.061}$	$0.621_{\pm 0.013}$	0.894
	DWF	$14.31_{\pm 7.26}$	$0.432_{\pm 0.041}$	$0.550_{\pm 0.015}$	0.467
	DWF†	$64.57_{\pm 7.53}$	$0.541_{\pm 0.061}$	$0.676_{\pm 0.018}$	0.795
	DWF-L	$14.06_{\pm 6.05}$	$0.416_{\pm 0.047}$	$0.521_{\pm 0.015}$	0.456
	DWF-L†	$59.62_{\pm 8.07}$	$0.512_{\pm 0.056}$	$0.638_{\pm 0.013}$	0.760

E Domain-Conditioned Generation

We study geographic distribution shift on Folktables ACSIncome (Ding et al., 2021), predicting whether an individual's annual income exceeds \$50,000 in California and Puerto Rico. These domains represent different regional populations under a shared prediction task. This models a setting where domain-specific classifiers are available but their original training data cannot be accessed or pooled. By learning a conditional distribution over their weights, we test whether unseen conditioning values can generate individual networks that balance predictive performance across both domains.

The flow is trained only on California (CA, $c = 0$) and Puerto Rico (PR, $c = 1$) classifiers. Training collections vary initialization, learning rate, and training duration, providing a control for mixtures of training recipes (Remark 3). Table 13 summarizes the data, classifier collections, conditional flow, and AUC evaluation settings.

F Heterogeneous Architecture Generation

We use one conditional flow to generate networks across 20 datasets and 16 native architectures, and to transfer to unseen hidden-width configurations. All generations use the same learned flow parameters and checkpoint, with no padding or architecture-specific learned parameters.

F.1 Data and Training Collections

We use 5K independently trained MLPs per OpenML dataset (Bischl et al., 2021), for 100K networks in total. The native architectures have one or two hidden layers, input dimensions from 4 to 216, and output dimensions from 2 to 26 (Table 15). Each dataset uses a stratified $80/20$ training–test

Table 13: ACS data, training collections, conditional flow, and AUC evaluation settings.

Data and training collections	
Data source	Folktables ACSIncome (Ding et al., 2021); 2018 one-year ACS data
Domains and split	CA ($c = 0$), PR ($c = 1$); up to 30K examples per domain; stratified $80/20$ training/test split
Categorical features	One-hot encode `COW`, `MAR`, `RELP`, `SEX`, and `RAC1P`
Numerical features	Median-impute and standardize `AGEP`, `SCHL`, and `WKHP`
Preprocessing	Fitted jointly on both training sets, then frozen; 44 input features
Collection size	5K independently trained networks per domain; default PyTorch initialization; final checkpoints only
Classifier architecture	MLP $44 \to 32 \to 32 \to 2$; 2,562 parameters
Classifier optimization	Adam; cross-entropy loss; batch size 256
Learning rate / epochs	Per network: log-uniform rate in $[3 \times 10^{-4}, 3 \times 10^{-3}]$; uniform integer in $[15, 40]$ epochs
Conditional flow	
Training conditions	$c \in \{0, 1\}$ only
Message-passing blocks	8
Parameters (M)	Approximately 7.8
Node / edge / time dimensions	128 / 128 / 128
Positional encodings	Orbit-shared
Message / global aggregation	Attention-weighted / mean
Condition embedding	Two-layer SiLU MLP, dimension 128; concatenated with time embedding for per-block AdaLN
Coupling / time sampling	Independent Gaussian–target coupling; uniform time sampling
Weight normalization	Parameter-group
Training updates	Approximately 57K
Global batch size	256
Learning rate / schedule	2×10^{-4} / cosine decay
EMA decay / condition dropout	0.999 / 0.1; EMA weights used for sampling
Generation and AUC evaluation	
Generated networks	50 per condition, $c \in \{0, 0.4, 0.5, 0.6, 1\}$
Guidance scale	1.5 at endpoints; 4.0 at intermediate conditions
ODE solver / refinement	`dopri5`; no refinement
Evaluation	Positive-class ROC AUC on both held-out domains
Baseline pairs	50 CA–PR checkpoint pairs for all baselines
Mixing and reporting	$\alpha \in \{0, 0.1, \ldots, 1\}$; curves report mean AUC over the 50 pairs
Weight averaging	Direct or after Git Re-Basin (Ainsworth et al., 2022) with 1, 5, or 50 matching iterations
Logit ensembling	Average logits before softmax

split. Categorical features are imputed with the most frequent value and one-hot encoded; numerical features are median-imputed and standardized. Preprocessing is fitted only on the training split. Source networks use ReLU after each hidden layer and retain only their final checkpoints.

F.2 Shared Representation and Conditioning

We represent each MLP as a graph following Lim et al. (2024). One permutation-equivariant GMN parameterizes the velocity field for every architecture: all encoders, message-passing blocks, and readouts share their learned parameters, without any architecture specific parameters or padding. Sinusoidal positional encodings use structural roles shared within hidden-unit permutation orbits.

Architecture is specified by the graph, while dataset identity c is linearly encoded and added to the time embedding for AdaLN conditioning. Dropped conditions use a learned null token. Four native architectures are each shared by two datasets, so architecture and task identity are supplied separately. Table 14 summarizes training and sampling.

Table 14: Heterogeneous collection construction, shared flow training, and evaluation. All datasets and width configurations use the same flow checkpoint and sampling settings. All the networks in the training collections were initialized with different seeds.

Training collections	
Collection size	100K networks; 20 datasets; 16 native architectures
Classifier optimization	Adam; learning rate 10^{-3}; cross-entropy loss
Training recipe	Batch size / epochs: $128/25$; $32/300$ for balance-scale, hayes-roth, and solar-flare (OpenML 40686)
Conditional flow	
Message-passing blocks	10
Parameters (M)	15.13
Node / edge / time dimensions	128 / 192 / 128
Activation / edge encoder	Tanh-approximate GELU / Fourier features
Weight normalization	None
Coupling / time sampling	Independent Gaussian–target coupling; uniform time sampling
Optimizer	AdamW; learning rate 5×10^{-4}; weight decay 0
Learning-rate schedule	2K-update linear warmup, then cosine decay over 1M scheduled updates
Training updates	917,600
Global batch size	192 across 6 GPUs
Gradient clipping	Global norm 1.0
EMA decay / condition dropout	0.999 / 0.1
Evaluation	
Generated networks	100 per dataset and width configuration
Hidden-width multipliers	0.5, 1.0, and 1.5
Guidance scale	1.0
ODE solver	`dopri5`
Scratch references	5K networks at native widths; 100 at each variant width, using the dataset's source recipe

F.3 Generation and Width Transfer

All networks are sampled using the conditional EMA velocity field with guidance scale 1.0. For zero-shot width transfer, we multiply every hidden width by 0.5 or 1.5, preserving depth and input/output dimensions. The resulting 32 distinct variant architectures are absent from the flow's training collection and require only a change to the input graph, without retraining or adding learned parameters. Dataset identities remain those seen during flow training. All results use three refinement cycles with $t^* = 0.9$, following Appendix C.4.

We compare test accuracy with independently trained networks at the same target architecture, evaluated on the same held-out examples. Native references contain all 5K source networks; each variant reference contains 100 networks trained with the dataset's source recipe in Table 14.

Results. Table 15 reports accuracies and gaps relative to training from scratch for all 60 dataset–width configurations. Averaging equally across datasets, generated networks trail the corresponding scratch references by 0.98 percentage points at native widths and $0.58/1.69$ points at unseen $0.5/1.5\times$ widths. At unseen $0.5\times$ widths, generation improves over scratch training on several datasets, including hayes-roth ($+14.03$ points) and mfeat-karhunen ($+5.12$ points), showing that width transfer can yield better-performing smaller networks. Performance varies across tasks, with a 25.16-point deficit on letter at this width. These results demonstrate generalization to unseen widths using the same flow parameters.

Table 15: Heterogeneous generation and width transfer. Headings give each dataset's native architecture; width multipliers apply to all hidden layers, with $0.5\times$ and $1.5\times$ unseen during flow training. Accuracy (%) is mean$_{\pm\text{std}}$; Δ is generated minus scratch accuracy in percentage points. All generated networks use one shared flow. Scratch references contain 5K native networks or 100 networks per variant; each generated entry uses 100 networks. All evaluations use the same held-out split per dataset. The final block averages all 20 datasets equally. Solar-flare (5 classes) is OpenML 40686.

Width	Scratch	Generated	Δ
kr-vs-kp $73 \to 32 \to 32 \to 2$			
$0.5\times$	$97.38_{\pm 0.72}$	$96.02_{\pm 1.62}$	-1.36
$1\times$	$98.61_{\pm 0.51}$	$97.62_{\pm 0.60}$	-0.99
$1.5\times$	$99.13_{\pm 0.39}$	$97.13_{\pm 0.94}$	-2.00
letter $16 \to 32 \to 32 \to 26$			
$0.5\times$	$79.54_{\pm 0.80}$	$54.38_{\pm 3.51}$	-25.16
$1\times$	$87.07_{\pm 0.50}$	$73.26_{\pm 1.82}$	-13.81
$1.5\times$	$90.45_{\pm 0.35}$	$73.96_{\pm 1.66}$	-16.49
balance-scale $4 \to 16 \to 3$			
$0.5\times$	$95.84_{\pm 1.82}$	$96.79_{\pm 1.32}$	$+0.95$
$1\times$	$96.96_{\pm 1.10}$	$96.83_{\pm 1.15}$	-0.13
$1.5\times$	$97.48_{\pm 0.91}$	$96.95_{\pm 1.18}$	-0.53
mfeat-factors $216 \to 32 \to 10$			
$0.5\times$	$96.17_{\pm 0.45}$	$94.78_{\pm 0.88}$	-1.39
$1\times$	$96.65_{\pm 0.37}$	$96.08_{\pm 0.48}$	-0.57
$1.5\times$	$96.80_{\pm 0.31}$	$96.39_{\pm 0.46}$	-0.41
mfeat-fourier $76 \to 64 \to 10$			
$0.5\times$	$83.22_{\pm 1.00}$	$83.15_{\pm 1.37}$	-0.07
$1\times$	$85.14_{\pm 0.87}$	$84.29_{\pm 1.03}$	-0.85
$1.5\times$	$85.86_{\pm 0.78}$	$84.29_{\pm 0.84}$	-1.57
breast-w $9 \to 64 \to 32 \to 2$			
$0.5\times$	$96.31_{\pm 0.28}$	$95.48_{\pm 0.81}$	-0.83
$1\times$	$96.28_{\pm 0.31}$	$96.21_{\pm 0.36}$	-0.07
$1.5\times$	$95.86_{\pm 0.48}$	$96.41_{\pm 0.12}$	$+0.55$
mfeat-karhunen $64 \to 16 \to 8 \to 10$			
$0.5\times$	$59.51_{\pm 7.43}$	$64.63_{\pm 6.25}$	$+5.12$
$1\times$	$89.04_{\pm 2.57}$	$87.02_{\pm 3.00}$	-2.02
$1.5\times$	$93.09_{\pm 0.95}$	$90.54_{\pm 1.39}$	-2.55
mfeat-morphological $6 \to 64 \to 32 \to 10$			
$0.5\times$	$71.55_{\pm 0.99}$	$72.01_{\pm 1.78}$	$+0.46$
$1\times$	$72.62_{\pm 0.84}$	$72.45_{\pm 1.25}$	-0.17
$1.5\times$	$73.87_{\pm 0.97}$	$71.69_{\pm 0.92}$	-2.18
mfeat-zernike $47 \to 24 \to 10$			
$0.5\times$	$73.41_{\pm 1.65}$	$75.97_{\pm 1.81}$	$+2.56$
$1\times$	$77.50_{\pm 1.26}$	$77.28_{\pm 1.18}$	-0.22
$1.5\times$	$79.15_{\pm 1.12}$	$77.67_{\pm 1.13}$	-1.48
cmc $24 \to 40 \to 20 \to 3$			
$0.5\times$	$54.94_{\pm 1.64}$	$55.71_{\pm 1.83}$	$+0.77$
$1\times$	$58.15_{\pm 1.28}$	$57.87_{\pm 1.37}$	-0.28
$1.5\times$	$58.82_{\pm 1.22}$	$58.12_{\pm 1.22}$	-0.70

Width	Scratch	Generated	Δ
blood-transfusion $4 \to 32 \to 2$			
$0.5\times$	$76.55_{\pm 1.24}$	$77.33_{\pm 1.06}$	$+0.78$
$1\times$	$77.49_{\pm 0.93}$	$77.46_{\pm 0.92}$	-0.03
$1.5\times$	$78.17_{\pm 0.74}$	$77.64_{\pm 0.89}$	-0.53
banknote-auth $4 \to 32 \to 2$			
$0.5\times$	$94.27_{\pm 2.13}$	$95.67_{\pm 3.27}$	$+1.40$
$1\times$	$96.38_{\pm 0.89}$	$96.24_{\pm 0.85}$	-0.14
$1.5\times$	$96.88_{\pm 0.41}$	$96.37_{\pm 0.71}$	-0.51
hayes-roth $4 \to 16 \to 3$			
$0.5\times$	$65.38_{\pm 6.46}$	$79.41_{\pm 6.93}$	$+14.03$
$1\times$	$81.52_{\pm 4.21}$	$82.22_{\pm 4.02}$	$+0.70$
$1.5\times$	$81.78_{\pm 3.78}$	$81.37_{\pm 3.48}$	-0.41
BNG(cmc) $24 \to 40 \to 20 \to 3$			
$0.5\times$	$57.57_{\pm 0.28}$	$54.52_{\pm 1.44}$	-3.05
$1\times$	$57.88_{\pm 0.29}$	$57.10_{\pm 0.48}$	-0.78
$1.5\times$	$57.83_{\pm 0.30}$	$57.18_{\pm 0.49}$	-0.65
heart-c $25 \to 32 \to 2$			
$0.5\times$	$78.93_{\pm 3.46}$	$76.97_{\pm 3.76}$	-1.96
$1\times$	$80.80_{\pm 2.53}$	$81.39_{\pm 2.12}$	$+0.59$
$1.5\times$	$81.34_{\pm 2.11}$	$80.56_{\pm 2.60}$	-0.78
heart-h $25 \to 32 \to 2$			
$0.5\times$	$75.61_{\pm 4.03}$	$79.15_{\pm 3.83}$	$+3.54$
$1\times$	$77.99_{\pm 3.28}$	$78.34_{\pm 3.69}$	$+0.35$
$1.5\times$	$80.42_{\pm 3.62}$	$77.85_{\pm 3.27}$	-2.57
ozone-level-8hr $72 \to 32 \to 2$			
$0.5\times$	$95.15_{\pm 0.31}$	$95.27_{\pm 0.39}$	$+0.12$
$1\times$	$95.46_{\pm 0.27}$	$95.40_{\pm 0.32}$	-0.06
$1.5\times$	$95.56_{\pm 0.26}$	$95.42_{\pm 0.28}$	-0.14
phoneme $5 \to 32 \to 2$			
$0.5\times$	$78.15_{\pm 0.68}$	$78.62_{\pm 0.88}$	$+0.47$
$1\times$	$78.88_{\pm 0.67}$	$78.78_{\pm 0.72}$	-0.10
$1.5\times$	$79.30_{\pm 0.71}$	$78.82_{\pm 0.58}$	-0.48
solar-flare (5 classes) $34 \to 24 \to 12 \to 5$			
$0.5\times$	$78.25_{\pm 2.59}$	$70.40_{\pm 5.83}$	-7.85
$1\times$	$75.44_{\pm 2.73}$	$74.37_{\pm 3.51}$	-1.07
$1.5\times$	$75.92_{\pm 2.18}$	$75.52_{\pm 2.93}$	-0.40
solar-flare (2 classes) $31 \to 32 \to 2$			
$0.5\times$	$83.01_{\pm 0.57}$	$82.97_{\pm 0.58}$	-0.04
$1\times$	$83.09_{\pm 0.59}$	$83.23_{\pm 0.60}$	$+0.14$
$1.5\times$	$83.00_{\pm 0.55}$	$83.05_{\pm 0.59}$	$+0.05$
Mean across all datasets			
$0.5\times$	79.54	78.96	-0.58
$1\times$	83.15	82.17	-0.98
$1.5\times$	84.04	82.35	-1.69