BEYOND MEMORIZATION: AUDITING AND GENERATING FUNCTIONAL NEURAL WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural weight generation promises to replace per-task optimization with reusable distribution learning over trained checkpoints, yet rigorous evaluation remains underdeveloped. Task accuracy alone cannot distinguish genuine synthesis from memorization and interpolation. Conversely, parameter-space novelty can arise from nonfunctional dispersion. We introduce WeightAUDIT, a five-metric diagnostic framework that jointly assesses parameter dependence, behavioral dependence, affine interpolation, task utility, and predictive diversity against independently trained references. Auditing nine generators reveals three dominant failure signatures, namely parameter memorization, interpolation, and ineffective generation. We also introduce Weight Diffusion with Instantiated Network Supervision (WINS). It instantiates complete networks from predicted clean weights and backpropagates task loss through these networks to train the denoiser. On KMNIST, WINS achieves 76.9% accuracy, exceeding the training mean of 72.8%, while maintaining weight novelty and predictive diversity near the reference level. Across MNIST, EMNIST, SVHN and CIFAR10, WINS retains utility without severe checkpoint contraction, interpolation, or predictive collapse, although performance varies by dataset. Varying the checkpoint pool size reveals an accuracy dip, followed by higher utility and lower training dependence in larger pools.

1 INTRODUCTION

Neural network weights are overwhelmingly produced by running tens of thousands of gradient descent iterations for most tasks. As public repositories now host millions of trained checkpoints (Schürholt et al., 2022b; Wang et al., 2026), an appealing alternative is to generate high-performing checkpoints directly from a learned distribution rather than training anew.

Recent work pursues this goal through two strategies: trajectory-based methods that approximate optimization paths (Peebles et al., 2022), and direct-solution methods that model populations of converged checkpoints (Schürholt et al., 2022a; Erkoç et al., 2023; Wang et al., 2024a; Lipman et al., 2023; Gupta et al., 2026). Conditional variants add task-level guidance (Soro et al., 2025; Li et al., 2024; Tian et al., 2025), and hypernetworks generate weights from conditioning inputs or latent noise (Ha et al., 2017; Charakorn et al., 2025; Ratzlaff & Li, 2019).

Task accuracy alone cannot distinguish genuine synthesis from memorization or interpolation. On the other hand, novelty alone is also insufficient because checkpoints far from the training pool may still fail on the target task. Existing diagnostics reveal these failures (Zeng et al., 2026), but differences in tasks, architectures, and checkpoint pools complicate comparisons across generators. These challenges motivate a common framework jointly assessing three distribution-level properties: (i) **Novelty** relative to the training pool, (ii) **Functional competence**, and (iii) **Behavioral diversity**.

We introduce **WeightAUDIT**, a common diagnostic framework with five explicitly defined metrics and reference normalization. Its profile distinguishes parameter memorization, behavioral memorization, interpolation, ineffective generation, and predictive collapse, which accuracy alone cannot detect. Fig. 1 illustrates genuine generation and these five failure signatures. Our audit of nine representative generators finds at least one diagnosed failure in every baseline, most commonly parameter memorization, interpolation, or ineffective generation. Under a controlled common setting, these failures fall into two regimes: novel but nonfunctional and functional but dependent on training checkpoints.

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.

We further propose **WINS**, a Weight Diffusion framework with Instantiated Network Supervision. Standard diffusion objectives model weight distributions and constrain predicted parameters near the training pool without ensuring a valid task mapping. WINS instantiates predicted clean weights as executable networks and backpropagates task cross-entropy through those weights, directly supervising the implemented function. This behavioral supervision enables novel, functional checkpoints in compact settings.

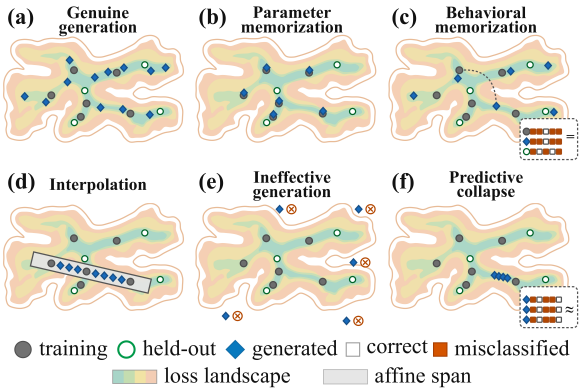


Figure 1: WeightAUDIT patterns (a)–(f) are genuine generation, parameter memorization, behavioral memorization, interpolation, ineffective generation, and predictive collapse. Markers indicate training (gray), held-out (green), and generated (blue) checkpoints. Contours show the loss landscape. The shaded band marks the training pool's affine span.

On KMNIST, WINS achieves 76.9% accuracy, exceeding the training mean of 72.8%, while maintaining weight novelty and predictive diversity near the reference level. Once trained, WINS generates 256 checkpoints in 1.96 seconds, compared with 35.4 minutes to train the same number using gradient descent with backpropagation. Varying the checkpoint pool size reveals an intermediate accuracy dip as training dependence decreases, followed by improved utility at larger pool sizes. This pattern motivates evaluating functionality and training dependence jointly.

Our contributions are threefold:

- We introduce WeightAUDIT, which jointly assesses novelty, functional competence, and behavioral diversity through five complementary metrics, enabling comparisons of weight generators beyond accuracy alone.
- We propose WINS, a weight diffusion model that jointly optimizes weight reconstruction and task performance by backpropagating task loss through complete networks instantiated from its clean-weight predictions.
- We conducted extensive experiments and separate sweeps of behavioral supervision strength and checkpoint pool size illustrate why generator evaluation should jointly track utility, predictive diversity, and training dependence.

2 Related Work

2.1 Evaluation of Generative Models

In data space, replication studies identify near-duplicates of training samples (Carlini et al., 2023; Somepalli et al., 2023), statistical tests detect copying against held-out data (Meehan et al., 2020), and fidelity and diversity metrics assess distributions (Sajjadi et al., 2018; Alaa et al., 2022). Diffusion models shift from memorization to generalization as training data grow (Kadkhodaie et al., 2024; Zhang et al., 2024b; Gu et al., 2025; Maillard & Goldt, 2026). We examine how training dependence, utility, and diversity change with checkpoint pool size in weight space. Zeng et al. (2026) diagnose weight generators using distances to the nearest training checkpoint, overlap between error sets, noise controls matched by behavioral similarity, and interpolation tests. WeightAUDIT builds on these diagnostics with explicit reference normalization, an affine-span residual, and within-population predictive disagreement. It combines these measurements into a joint profile of training dependence, interpolation, utility, and predictive diversity.

2.2 Weight-space Structure

Weight-space generation must also contend with geometry. Under neuron permutations, a single function admits combinatorially many parameterizations, and independently trained solutions are often linearly connected once these permutations are aligned (Hecht-Nielsen, 1990; Garipov et al.,

2018; Entezari et al., 2022; Ainsworth et al., 2023). Converged solutions also concentrate near structures of low intrinsic dimension (Li et al., 2018; Mao et al., 2024; Wang et al., 2024b). One line of work handles the symmetry with permutation-equivariant architectures that process weights as structured inputs (Navon et al., 2023; Zhou et al., 2023; Kofinas et al., 2024). WINS instead canonicalizes checkpoints into a shared basin, which lets a standard transformer denoiser model complete weight vectors. Our ablation studies quantify the contribution of this alignment.

2.3 Weight Generation Methods

Existing approaches differ in their training data and conditioning signals (Han et al., 2026). G.pt learns updates from optimization trajectories (Peebles et al., 2022), while MetaDiff denoises classifier weights conditioned on support gradients (Zhang et al., 2024a). Direct-solution methods model populations of converged checkpoints through autoencoder-based hyper-representations (Schürholt et al., 2022a), diffusion over weights (Erkoç et al., 2023; Wang et al., 2024a), and flow matching (Lipman et al., 2023; Gupta et al., 2026), with extensions to larger architectures (Schürholt et al., 2024; Wang et al., 2025). Conditional variants add task-level guidance (Soro et al., 2025; Li et al., 2024; Tian et al., 2025). Hypernetworks generate weights from conditioning inputs (Ha et al., 2017; Charakorn et al., 2025). HyperGAN trains generators of complete networks with task loss and adversarial regularization of latent codes. The variability in its generated weight norms decreases during later training (Ratzlaff & Li, 2019). Deutsch et al. (2019) train hypernetworks using task loss and entropy regularization. Our audit jointly examines task performance, training dependence, interpolation, and predictive diversity.

3 WeightAUDIT: A Diagnostic Framework for Weight Generation

WeightAUDIT evaluates whether generated checkpoints depart from the training pool while retaining task utility and predictive diversity. Its five complementary metrics quantify parameter dependence, behavioral dependence, affine confinement, utility gain, and predictive disagreement.

3.1 Evaluation Setup

For a fixed task and target architecture, let Θ_{tr}, Θ_{ho}, and Θ_{gen} denote the training, held-out, and generated checkpoint sets, respectively. We use θ for the corresponding neural network weight, f_θ for the network instantiated from that weight, and $\widehat{y}_\theta(x) = \arg\max f_\theta(x)$ for its predicted class.

3.2 Metric Definitions

(i) Weight Dependence Gap measures the degree to which generated parameter vectors contract toward those of the training checkpoints in parameter space:

$$M_{\mathrm{WDG}} = \frac{\mathbb{E}_{\theta\sim\Theta_{\mathrm{gen}}}[d_{\mathrm{tr}}(\theta)]}{\mathbb{E}_{\theta\sim\Theta_{\mathrm{ho}}}[d_{\mathrm{tr}}(\theta)]}, \tag{1}$$

where $d_{\mathrm{tr}}(\theta) = \min_{\theta^{(j)}\in\Theta_{\mathrm{tr}}} \|\theta - \theta^{(j)}\|_2$ is the nearest-neighbor distance from θ to the training set.

$M_{\mathrm{WDG}} \approx 1$ indicates separation from the training pool comparable to held-out checkpoints, and $M_{\mathrm{WDG}} \ll 1$ reveals memorization (Zeng et al., 2026).

(ii) Behavior Dependence Gap checks whether the error patterns of generated checkpoints differ from those of training checkpoints. For f_θ, let $E(\theta) = \{(x, y) \in \mathcal{D}_{\mathrm{eval}} : \widehat{y}_\theta(x) \neq y\}$ denote its error set on evaluation data. The behavioral similarity between θ and the training checkpoint pool is $s(\theta) = \max_{\theta^{(j)}\in\Theta_{\mathrm{tr}}} \frac{|E(\theta)\cap E(\theta^{(j)})|}{|E(\theta)\cup E(\theta^{(j)})|}$. BDG is defined as:

$$M_{\mathrm{BDG}} = \frac{1 - \mathbb{E}_{\theta\sim\Theta_{\mathrm{gen}}}[s(\theta)]}{1 - \mathbb{E}_{\theta\sim\Theta_{\mathrm{ho}}}[s(\theta)]}. \tag{2}$$

$M_{\mathrm{BDG}} \approx 1$ indicates comparable separation of generated and held-out checkpoints from training error patterns, whereas $M_{\mathrm{BDG}} \ll 1$ reveals replication of these patterns (Zeng et al., 2026; Meynent et al., 2025).

(iii) Interpolation Exclusion further rules out affine interpolation traps in which a generated parameter vector may be far from every training vector yet still lie within their affine span (Izmailov et al., 2018; Wortsman et al., 2022). Let $\mu_{\mathrm{tr}} = \frac{1}{N_{\mathrm{tr}}} \sum_{\theta^{(i)} \in \Theta_{\mathrm{tr}}} \theta^{(i)}$ be the training mean and $\mathcal{S}_{\mathrm{tr}} = \mu_{\mathrm{tr}} + \mathrm{span}\{\theta^{(i)} - \mu_{\mathrm{tr}} : \theta^{(i)} \in \Theta_{\mathrm{tr}}\}$ the affine span. For a weight vector θ, the normalized off-span residual is $r(\theta) = \frac{\|\theta - \Pi_{\mathcal{S}_{\mathrm{tr}}}(\theta)\|_2}{\|\theta - \mu_{\mathrm{tr}}\|_2}$. IX is defined as:

$$M_{\mathrm{IX}} = \frac{\mathbb{E}_{\theta \sim \Theta_{\mathrm{gen}}}[r(\theta)]}{\mathbb{E}_{\theta \sim \Theta_{\mathrm{ho}}}[r(\theta)]}. \tag{3}$$

$M_{\mathrm{IX}} \approx 1$ indicates off-span deviation comparable to that of held-out checkpoints, whereas smaller values indicate stronger confinement to the training affine span. Throughout, *interpolation* denotes this confinement, including extrapolative affine recombinations of training checkpoints.

(iv) Utility Gain examines whether this novelty translates into functional utility rather than destructive random perturbation. It compares the accuracy of generated checkpoints with that of a noise baseline matched to their behavioral separation from the training checkpoint pool (Zeng et al., 2026).

$$M_{\mathrm{UG}} = \frac{A_{\mathrm{gen}} - A_{\mathrm{noise}}}{A_{\mathrm{tr}} - A_{\mathrm{noise}}}, \tag{4}$$

where A_{gen}, A_{tr}, and A_{noise} denote the average accuracies obtained from generated checkpoints, training checkpoints, and matched-noise parameter vectors, respectively.

$M_{\mathrm{UG}} \approx 1$ indicates utility comparable to the training pool, and values near zero indicate ineffective generation. Negative values indicate accuracy below the matched noise baseline.

(v) Predictive Diversity measures mean pairwise predictive disagreement within Θ_{gen}, normalized by the corresponding disagreement within Θ_{tr} (Lakshminarayanan et al., 2017; Fort et al., 2019).

$$M_{\mathrm{PD}} = \frac{\mathrm{Div}(\Theta_{\mathrm{gen}})}{\mathrm{Div}(\Theta_{\mathrm{tr}})}, \tag{5}$$

where $\mathrm{Div}(\Theta) = \frac{2}{|\Theta|(|\Theta|-1)} \sum_{i<j} \delta(\theta^{(i)}, \theta^{(j)})$ with $\Theta = \{\theta^{(i)}\}_{i=1}^{|\Theta|}$, $\delta(\theta^{(i)}, \theta^{(j)}) = \frac{1}{|\mathcal{D}_{\mathrm{eval}}|} \sum_{(x,y)} \mathbb{I}[\widehat{y}_{\theta^{(i)}}(x) \neq \widehat{y}_{\theta^{(j)}}(x)]$ denoting the predictive disagreement rate between the two instantiated networks.

$M_{\mathrm{PD}} \approx 1$ indicates diversity comparable to that of the training pool, whereas $M_{\mathrm{PD}} \ll 1$ reveals predictive collapse.

Note that values above one indicate greater behavioral separation (BDG), larger normalized affine residuals (IX), or greater predictive disagreement (PD) relative to their references, but do not by themselves imply better generation. Appendix D.8 details metric bounds and thresholds.

4 WINS: Weight Diffusion with Instantiated Network Supervision

Functional neural network weights occupy only a small subset of weight space, so parameter-space novelty or distributional fidelity alone does not ensure valid generation (Garipov et al., 2018; Li et al., 2018; Entezari et al., 2022; Mao et al., 2024). WINS (Fig. 2) combines Symmetry-Consistent Weight Representation (SWR) and Tokenized Weight Diffusion (TWD) for weight-space modeling, with Differentiable Behavioral Supervision (DBS) providing task supervision.

4.1 Symmetry-Consistent Weight Representation

Neuron permutations induce an exponentially large equivalence class of functionally identical networks (Hecht-Nielsen, 1990; Entezari et al., 2022), scattering equivalent solutions across distant regions of parameter space. To remove redundancy, WINS aligns each training checkpoint to a common reference checkpoint θ_{ref} via a weight-matching operator $\tilde{\theta}^{(i)} = \mathcal{A}(\theta^{(i)}; \theta_{\mathrm{ref}})$, where $\mathcal{A}$ reorders hidden channels while preserving the network mapping $f_{\theta^{(i)}}(x) = f_{\tilde{\theta}^{(i)}}(x)$. We implement $\mathcal{A}$ using Git Re-Basin (Ainsworth et al., 2023).

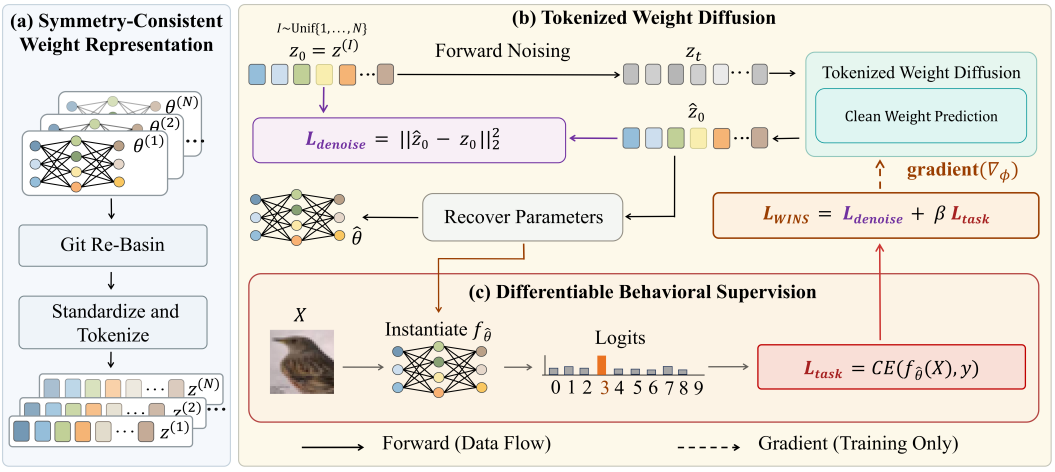


Figure 2: WINS overview. (a) Git Re-Basin aligns training checkpoints to a shared reference before standardization and tokenization. (b) A Transformer predicts the clean standardized vector $\hat{z}_0$ from z_t and inverse standardization recovers $\hat{\theta}$. (c) Instantiating $f_{\hat{\theta}}$ enables task-loss backpropagation through $\hat{\theta}$ to the Transformer, directly supervising the implemented function.

The aligned checkpoints share a consistent parameter coordinate system, enabling meaningful distribution modeling in the quotient space modulo permutation symmetries. Since parameter dimensions span heterogeneous numerical scales, WINS further computes element-wise mean μ and standard deviation σ over the aligned training pool, and standardizes weights as $z^{(i)} = (\tilde{\theta}^{(i)} - \mu)/\sigma$. Generation uses this standardized space. Inverse standardization restores the original parameter scale.

4.2 Tokenized Weight Diffusion

Modeling complete weight vectors in high-dimensional space requires structured handling of cross-parameter dependencies. Let $I \sim \mathrm{Unif}\{1, \ldots, N\}$ be a random checkpoint index and set $z_0 = z^{(I)}$, where $\mathcal{Z}_{\mathrm{tr}} = \{z^{(i)}\}_{i=1}^{N}$ is the standardized aligned checkpoint pool. At step t, each diffusion state $z_t \in \mathbb{R}^D$ is partitioned into a sequence of fixed-size contiguous blocks that are positionally encoded, processed by a Transformer (Vaswani et al., 2017), and decoded back to parameter blocks. A Gaussian diffusion process is applied (Ho et al., 2020; Song et al., 2021b):

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \tag{6}$$

where $\epsilon \sim \mathcal{N}(0, I_D)$ is a Gaussian noise vector and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative noise schedule.

Crucially, the TWD denoiser $g_\phi(\cdot)$ directly predicts the clean standardized state rather than the noise (Ho et al., 2020; Karras et al., 2022):

$$\hat{z}_0 = g_\phi(z_t, t), \tag{7}$$

where $g_\phi(\cdot)$ is a Transformer parameterized by ϕ.

Direct prediction of z_0 provides a complete estimate of the clean state at every diffusion step. After parameter recovery, this estimate yields an executable network that can be evaluated with a behavioral loss. The denoising objective is:

$$\mathcal{L}_{\mathrm{denoise}} = \mathbb{E}_{z_0, t, \epsilon}\left[\frac{1}{D}\|\hat{z}_0 - z_0\|_2^2\right]. \tag{8}$$

At inference, WINS initializes $z_T^{\mathrm{gen}} \sim \mathcal{N}(0, I_D)$ at step T and applies iterative reverse diffusion to obtain z_0^{gen}, then recovers $\theta^{\mathrm{gen}} = \mu + \sigma \odot z_0^{\mathrm{gen}}$ and instantiates the executable network $f_{\theta^{\mathrm{gen}}}$.

4.3 Differentiable Behavioral Supervision

The denoising objective in Eqn. (8) constrains predicted parameters near the standardized training distribution without ensuring functionality (Meynent et al., 2025). Given z_t, DBS recovers the

complete parameter vector $\hat{\theta}$ from $\hat{z}_0$ and instantiates $f_{\hat{\theta}}$. For a minibatch $\mathcal{B} = \{(x_b, y_b)\}_{b=1}^{B}$ from the training set, the behavioral loss is:

$$\mathcal{L}_{\text{task}} = \frac{1}{B} \sum_{b=1}^{B} \text{CE}\big(f_{\hat{\theta}}(x_b), y_b\big) \,. \tag{9}$$

Task gradients backpropagate to the denoiser parameters ϕ through the instantiated network's forward computation and differentiable parameter recovery. For $\mathbf{X} = (x_1, \ldots, x_B)$, let $f_{\hat{\theta}}(\mathbf{X})$ denote the minibatch logits. Formally:

$$\frac{\partial \mathcal{L}_{\text{task}}}{\partial \phi} = \underbrace{\frac{\partial \mathcal{L}_{\text{task}}}{\partial f_{\hat{\theta}}(\mathbf{X})}}_{\text{task loss}} \cdot \underbrace{\frac{\partial f_{\hat{\theta}}(\mathbf{X})}{\partial \hat{\theta}}}_{\text{network weights}} \cdot \underbrace{\frac{\partial \hat{\theta}}{\partial \hat{z}_0}}_{\text{parameter recovery}} \cdot \underbrace{\frac{\partial \hat{z}_0}{\partial \phi}}_{\text{denoiser}} \,. \tag{10}$$

It directly supervises the function implemented by the weights by binding weights to the target architecture's computation graph, not merely their location in parameter space.

Training Objective. WINS combines two loss terms:

$$\mathcal{L}_{\text{WINS}} = \mathcal{L}_{\text{denoise}} + \beta \mathcal{L}_{\text{task}}, \tag{11}$$

where $\beta \geq 0$ controls the strength of behavioral supervision.

The denoising loss $\mathcal{L}_{\text{denoise}}$ maintains fidelity to the checkpoint distribution, while the behavioral loss $\mathcal{L}_{\text{task}}$ preserves functionality. Behavioral supervision contributes to the total update $\nabla_\phi \mathcal{L}_{\text{WINS}}$ through gradients backpropagated across the recovered parameter vector $\hat{\theta}$.

5 Experiments

In this section, we first audit existing generators in their original settings, then compare compatible methods under a shared checkpoint pool, finally examine how WINS behaves across datasets, architectures, supervision strengths, checkpoint pool sizes, and component choices.

Setup. By default, the following setup applies to Protocol II and the WINS experiments. We evaluate MNIST (LeCun et al., 1998), EMNIST (Cohen et al., 2017), KMNIST (Clanuwat et al., 2018), SVHN (Netzer et al., 2011), and CIFAR10 (Krizhevsky, 2009) using TinyCNN with $D = 6{,}738$ trainable parameters. For the architecture comparison, we additionally evaluate TinyCNN configurations with $D = 30{,}634$ and $D = 113{,}738$, as reported in Table 4.

Checkpoint construction: For each dataset, we train 3,256 target networks with distinct seeds, independent initialization and data shuffling, using Adam (Kingma & Ba, 2015) with learning rate 0.001 for 80 epochs. We retain the final weights and randomly split the checkpoints into $3{,}000$ for generator training and 256 references.

Evaluation: Each generator produces 256 checkpoints. We report WeightAUDIT profiles and mean accuracy. By default, experiments use five independent generator seeds. Tables report seed means.

Baseline Methods. We compare HyperDiffusion (Erkoç et al., 2023), DeepWeightFlow (Gupta et al., 2026), Hyper-Representations (Schürholt et al., 2022a), RPG (Wang et al., 2025), p-diff (Wang et al., 2024a), Tina (Li et al., 2024), D2NWG (Soro et al., 2025), G.pt (Peebles et al., 2022), and Text2Weight (Tian et al., 2025).

Evaluation Protocols. **Protocol I (Original-Setting Audit)** evaluates nine generators in their own optimized settings, retaining their original tasks, architectures, and checkpoint pools. **Protocol II (Same-Setting Comparison)** retrains seven compatible generators[1] on an identical KMNIST checkpoint pool to isolate method effects. Protocol I diagnoses intrinsic failures in optimized settings. Protocol II tests their persistence under matched conditions.

[1] G.pt requires optimization trajectories rather than converged checkpoints. Text2Weight generates only classification heads on frozen backbones from cross-task text descriptions. Neither supports our pool of complete checkpoints for a single target architecture.

Table 1: WeightAUDIT results under each method's original setup (Protocol I). Light gray marks moderate degradation ($0.40 \leq M < 0.70$) and dark gray severe failure ($M < 0.40$). Dashes denote unavailable measurements.

Method	Gen.	Train	WDG	BDG	IX	UG	PD
HyperDiffusion (2023)	–	–	0.06	–	0.53	–	2.14
DeepWeightFlow (2026)	0.961	0.963	0.08	0.07	0.21	0.31	1.06
Hyper-Representations (2022a)	0.679	0.832	0.20	0.96	1.01	0.71	1.37
RPG (2025)	0.908	0.937	5.98	35.92	0.10	0.66	0.63
p-diff (2024a)	0.750	0.770	17.91	6.29	0.26	0.78	0.53
Tina (2024)	0.644	0.674	1.71	3.11	1.14	0.16	0.60
D2NWG (2025)	0.928	0.937	1.99	2.66	1.07	0.26	0.41
G.pt (2022)	0.940	0.945	0.61	0.85	–	0.37	1.14
Text2Weight (2025)	0.654	0.681	2.41	1.44	1.05	0.35	0.43

Table 2: WeightAUDIT profiles of compatible baselines and WINS trained and tested on the same KMNIST checkpoint pool (Protocol II). Shading follows Table 1.

Method	Gen.	Train	WDG	BDG	IX	UG	PD
p-diff (2024a)	0.127	0.728	0.88	2.10	1.39	−0.04	4.11
Hyper-Representations (2022a)	0.094	0.728	0.82	2.14	1.03	−0.10	1.38
Tina (2024)	0.126	0.728	0.96	2.08	1.28	−0.10	2.06
RPG (2025)	0.117	0.728	1.39	2.11	2.17	−0.06	4.20
D2NWG (2025)	0.299	0.728	0.74	1.93	0.97	−0.16	3.72
HyperDiffusion (2023)	0.704	0.728	0.18	0.60	0.24	0.77	1.24
DeepWeightFlow (2026)	0.689	0.728	0.41	0.78	0.52	0.62	1.39
WINS (ours)	**0.769**	0.728	0.96	1.72	1.62	1.16	0.99

5.1 Original-Setting Audit

Table 1 presents WeightAUDIT profiles of nine generators in their default settings. The results reveal three dominant failure signatures:

Parameter memorization (WDG $\ll$ 1). HyperDiffusion, DeepWeightFlow, and Hyper-Representations contract strongly toward the training pool in parameter space. Their accuracy, where reported, coexists with strong contraction toward the training pool.

Interpolation (IX $\ll$ 1). RPG and p-diff remain largely confined to the affine span of the training checkpoints, even while far from every individual checkpoint (WDG $\gg$ 1).

Ineffective generation (UG $\ll$ 1). Tina, D2NWG, G.pt, and Text2Weight show limited normalized accuracy gains over the matched noise baseline, often accompanied by reduced predictive diversity.

These failures persist even in each method's own optimized setting. Protocol II next tests whether they persist under controlled conditions.

5.2 Same-Setting Comparison

Table 2 reports WeightAUDIT profiles of seven compatible generators retrained on the same pool of 3,000 KMNIST checkpoints, revealing two regimes.

Novel-but-nonfunctional dispersion. Tina, Hyper-Representations, p-diff, RPG, and D2NWG achieve parameter novelty comparable to held-out checkpoints and high predictive diversity, but accuracy collapses to 0.094–0.299 with UG ≤ 0, indicating nonfunctional generated weights.

Functional but dependent on training checkpoints. HyperDiffusion and DeepWeightFlow reach accuracies of 0.704 and 0.689 but remain close to the training pool (WDG ≤ 0.41, IX ≤ 0.52, BDG ≤ 0.78). Both fall below the training mean of 0.728 (UG = 0.77 and 0.62), combining useful task performance with substantial training dependence.

These profiles distinguish task performance from training dependence and predictive diversity. WINS attains accuracy 0.769 against the training mean of 0.728, with WDG 0.96 and PD 0.99 indicating parameter separation and predictive diversity near their reference levels.

Comparison across Protocols. HyperDiffusion and DeepWeightFlow remain dependent on training checkpoints in both protocols. In contrast, p-diff and RPG shift from interpolation in Protocol I to nonfunctional dispersion in Protocol II, showing that diagnostic signatures depend on the setting.

5.3 WINS ACROSS DATASETS AND ARCHITECTURES

Table 3 evaluates WINS on five datasets with the default TinyCNN ($D = 6{,}738$) and common setup. MNIST and SVHN mean accuracies closely match or exceed their training means (0.933 vs. 0.934 and 0.695 vs. 0.685), with WDG, IX, UG, and PD near or above reference levels. KMNIST also exceeds its training mean. EMNIST and CIFAR10 have lower utility (UG 0.85 and 0.81). WINS avoids parameter and behavioral memorization, interpolation, and predictive collapse on all datasets, but utility varies.

Table 3: Evaluating WINS across different datasets.

Dataset	Gen.	Train	WDG	BDG	IX	UG	PD
MNIST	0.933	0.934	1.09	2.61	0.99	0.99	1.60
KMNIST	0.769	0.728	0.96	1.72	1.62	1.16	0.99
EMNIST	0.936	0.951	0.99	1.63	1.04	0.85	3.11
SVHN	0.695	0.685	1.05	1.40	1.16	1.02	1.07
CIFAR10	0.555	0.603	0.88	1.25	1.26	0.81	0.92

Once trained for a fixed task and architecture, WINS generates 256 complete TinyCNN checkpoints in 1.96 s on an RTX 5090, compared with 35.4 min for 256 independent training runs. This comparison measures generation after offline preparation. The appendix reports checkpoint construction and generator training costs separately.

Table 4: WINS scalability to larger parameter counts.

Dataset	Params.	Gen.	Train	WDG	BDG	IX	UG	PD
SVHN	6,738	0.695	0.685	1.05	1.40	1.16	1.02	1.07
CIFAR10	6,738	0.555	0.603	0.88	1.25	1.26	0.81	0.92
SVHN	30,634	0.788	0.810	1.10	1.20	1.27	0.91	1.60
CIFAR10	30,634	0.672	0.727	0.89	1.43	1.17	0.74	1.21
SVHN	113,738	0.851	0.908	1.03	1.34	1.00	0.85	1.87
CIFAR10	113,738	0.698	0.782	0.91	1.42	1.20	0.72	1.16

For larger targets on SVHN and CIFAR10 (Table 4), WINS again avoids all four failure signatures. Mean accuracy increases from 0.695 to 0.851 and 0.555 to 0.698, respectively, while UG decreases from 1.02 to 0.85 and 0.81 to 0.72. Thus, its profile remains healthy as generation becomes more challenging, with modestly lower relative utility.

5.4 ABLATION STUDY

In this section, we report the most representative metrics. Appendix reports the complete WeightAUDIT profiles.

Behavioral Supervision Strength. On KMNIST, we vary β from 0 to 2 (Fig. 3). Pure diffusion ($\beta = 0$) produces novel, dispersed checkpoints (WDG 1.06, IX 1.06, PD 4.07) but low accuracy (0.165) and utility (UG -0.02), showing that parameter novelty alone does not imply functionality. Moderate supervision ($\beta \in [0.4, 0.5]$) restores accuracy above the training mean while keeping PD near its reference level. Stronger supervision brings diminishing accuracy gains and steadily lowers PD, consistent with the fidelity–diversity tradeoff in data-space diffusion guidance (Dhariwal & Nichol, 2021). At $\beta = 1.9$, accuracy reaches 0.828 but PD falls to 0.28, favoring moderate supervision for balancing accuracy and diversity.

Checkpoint Pool Size. Fig. 4 reports the checkpoint pool size (N) sweep. For $N \leq 40$, WDG far below its reference level indicates strong dependence on training checkpoints. Between $N = 40$ and $N = 128$, WDG rises sharply as accuracy temporarily falls to 0.653, consistent with departure from

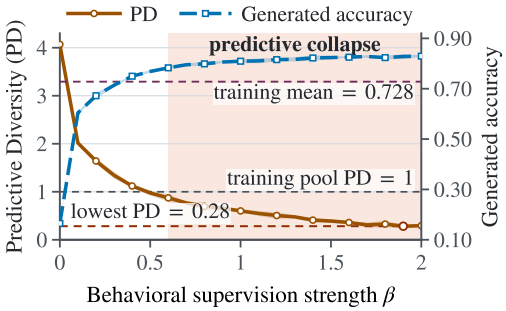


Figure 3: Trade-off between predictive diversity and generated checkpoint accuracy under varying β on KMNIST.

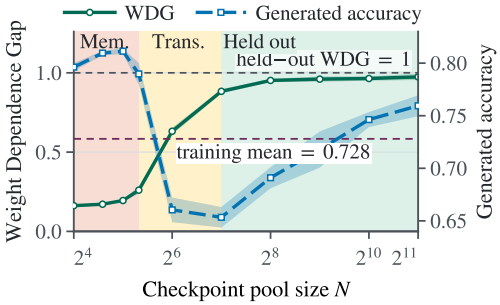


Figure 4: WINS accuracy and weight dependence with increasing checkpoint pool size N on KMNIST.

training-checkpoint modes and the memorization-to-generalization transition in data space (Kadkhodaie et al., 2024; Zhang et al., 2024b; Maillard & Goldt, 2026). Beyond $N = 128$, novelty and performance improve. WDG reaches 0.952 at $N = 256$ and stays near reference for larger N. Accuracy recovers from 0.691 to 0.759 at $N = 2048$ and exceeds the training mean for $N \geq 1024$.

Component Ablation Analysis. Table 5 compares ablations of the training objective, permutation alignment, and prediction target.

Training Objectives: Diffusion alone loses utility. CE alone preserves accuracy but collapses predictive diversity. Their combination provides the best observed utility–diversity balance in this KMNIST configuration.

Table 5: Component ablations on KMNIST. The training checkpoint mean accuracy is 0.728.

Variant	Gen.	WDG	BDG	IX	UG	PD
Full	0.769	0.96	1.72	1.62	1.16	0.99
Diff.-only	0.165	1.06	2.08	1.06	−0.02	4.07
CE-only	0.818	1.49	1.79	2.02	1.35	0.01
w/o align.	0.732	1.19	1.66	1.12	1.09	0.66
ϵ-pred.	0.519	1.60	1.81	1.85	0.25	2.72

Permutation Alignment: Removing alignment reduces accuracy from 0.769 to 0.732 and PD from 0.99 to 0.66, indicating that canonicalization facilitates this tradeoff.

Prediction Target: At the shared supervision coefficient, the variant that predicts ϵ achieves accuracy 0.519 and UG 0.25. This comparison favors direct z_0 prediction under the tested training configuration.

6 Conclusion

We introduced WeightAUDIT, a five-metric framework for jointly assessing weight novelty, functional utility, and predictive diversity. Auditing representative methods reveals that the generated neural network weights by baseline methods exhibit at least one failure signature, and under the controlled same-setting comparison, these failures consolidate into novel-but-nonfunctional and functional-but-training-dependent regimes. We further proposed WINS by combining tokenized weight diffusion with behavioral supervision through instantiated networks, where moderate supervision restores functionality while preserving diversity, but excessive supervision induces predictive collapse. Across checkpoint pool sizes, an intermediate accuracy dip precedes improved utility and reduced training dependence at larger pools. Together, these results provide a practical basis for distinguishing gains in task performance from progress in generating novel and behaviorally diverse networks. They also motivate diffusion training objectives that account for the functions implemented by generated weights, alongside diagnostics that expose training dependence and predictive collapse. These results concern compact networks. Future work will investigate how optimization and checkpoint diversity shape the accuracy dip and extend the analysis to larger architectures.

AI USE STATEMENT

Generative AI tools assisted with language editing, LaTeX formatting, and the formatting of existing code, organization of comments, and arrangement of files. The code assistance did not change the algorithmic logic. They were not used to design the study, implement the methods, conduct experiments, interpret results, or develop mathematical proofs. The authors take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

The WeightAUDIT framework and WINS method are described in Sections 3 and 4. Appendix A details the target architectures, datasets, checkpoint construction, and information isolation. Appendices D, E, and F provide metric implementation, training and sampling settings, and evaluation protocols. Complete diagnostic profiles and computational resources are reported in Appendices B and G.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Ahmed Alaa, Boris van Breugel, Evgeny S. Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 290–306. PMLR, 2022. URL https://proceedings.mlr.press/v162/alaa22a.html.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium*, pp. 5253–5270. USENIX Association, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/carlini.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7485–7514. PMLR, 2025. URL https://proceedings.mlr.press/v267/charakorn25a.html.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature, 2018. URL https://arxiv.org/abs/1812.01718.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017. URL https://ieeexplore.ieee.org/document/7966217.

Lior Deutsch, Erik Nijkamp, and Yu Yang. A generative model for sampling high-performance and diverse weights for neural networks, 2019. URL https://arxiv.org/abs/1905.02898.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021. URL https://papers.nips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. HyperDiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of*

the IEEE/CVF International Conference on Computer Vision, pp. 14300–14310, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Erkoc_HyperDiffusion_Generating_Implicit_Neural_Fields_with_Weight-Space_Diffusion_ICCV_2023_paper.html.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2019. URL https://arxiv.org/abs/1912.02757.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/be3087e74e9100d4bc4c6268cdbe8456-Abstract.html.

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=D3DBqvSDbj.

Saumya Gupta, Scott Biggs, Moritz Laber, Zohair Shafi, Robin Walters, and Ayan Paul. DeepWeightFlow: Re-basined flow matching for generating neural network weights. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fOwsr1VTi8.

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

Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, Rose Yu, Haggai Maron, Yanfang Ye, Lu Yin, and Ferrante Neri. A survey of weight space learning: Understanding, representation, and generation, 2026. URL https://arxiv.org/abs/2603.10090.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. North-Holland, 1990. URL https://www.sciencedirect.com/science/article/abs/pii/B9780444884008500194.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 876–885. AUAI Press, 2018. URL https://arxiv.org/abs/1803.05407.

Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cbaf319a4712385b5ba8a414808b5713-Abstract-Conference.html.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph neural networks for learning equivariant representations of neural networks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oO6FsMyDBt.

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.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

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. URL https://ieeexplore.ieee.org/document/726791.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryup8-WCW.

Zexi Li, Lingzhi Gao, and Chao Wu. Text-to-model: Text-conditioned neural network diffusion for train-once-for-all personalization, 2024. URL https://arxiv.org/abs/2405.14132.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Antoine Maillard and Sebastian Goldt. Memorisation, convergence and generalisation in generative models, 2026. URL https://arxiv.org/abs/2605.21402.

Jialin Mao, Itay Griniasty, Han Kheng Teoh, Rahul Ramesh, Rubing Yang, Mark K. Transtrum, James P. Sethna, and Pratik Chaudhari. The training process of many deep networks explores the same low-dimensional manifold. *Proceedings of the National Academy of Sciences*, 121(12): e2310002121, 2024. doi: 10.1073/pnas.2310002121. URL https://www.pnas.org/doi/10.1073/pnas.2310002121.

Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A three sample hypothesis test for evaluating generative models. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3546–3556. PMLR, 2020. URL https://proceedings.mlr.press/v108/meehan20a.html.

Léo Meynent, Ivan Melev, Konstantin Schürholt, Göran Kauermann, and Damian Borth. Structure is not enough: Leveraging behavior for neural network weight reconstruction. In *ICLR 2025 Workshop on Weight Space Learning*, 2025. URL https://openreview.net/forum?id=APsHrpqO3W.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25790–25816. PMLR, 2023. URL https://proceedings.mlr.press/v202/navon23a.html.

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

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021. URL https://proceedings.mlr.press/v139/nichol21a.html.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Neale Ratzlaff and Fuxin Li. HyperGAN: A generative model for diverse, performant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5361–5369. PMLR, 2019. URL https://proceedings.mlr.press/v97/ratzlaff19a.html.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/f7696a9b362ac5a51c3dc8f098b73923-Abstract.html.

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i-Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27906–27920, 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b2c4b7d34b3d96b9dc12f7bce424b7ae-Abstract-Conference.html.

Konstantin Schürholt, Diyar Taskiran, Boris Knyazev, Xavier Giró-i-Nieto, and Damian Borth. Model zoos: A dataset of diverse populations of neural network models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 35, 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/f94d5edb5c01715d879693ddbfdc1b98-Abstract-Datasets_and_Benchmarks.html.

Konstantin Schürholt, Michael W. Mahoney, and Damian Borth. Towards scalable and versatile weight space learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43947–43966. PMLR, 2024. URL https://proceedings.mlr.press/v235/schurholt24a.html.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? Investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6048–6058, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Somepalli_Diffusion_Art_or_Digital_Forgery_Investigating_Data_Replication_in_Diffusion_CVPR_2023_paper.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

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*, 2025. URL https://openreview.net/forum?id=j8WHjM9aMm.

Bowen Tian, Wenshuo Chen, Zexi Li, Songning Lai, Jiemin Wu, and Yutao Yue. Text2Weight: Bridging natural language and neural network weight spaces. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10152–10160, 2025. doi: 10.1145/3746027.3755441. URL https://doi.org/10.1145/3746027.3755441.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion, 2024a. URL https://arxiv.org/abs/2402.13144.

Kai Wang, Dongwen Tang, Wangbo Zhao, Konstantin Schürholt, Zhangyang Wang, and Yang You. Scaling up parameter generation: A recurrent diffusion approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://neurips.cc/virtual/2025/poster/119814.

Peng Wang, Huijie Zhang, Zekai Zhang, Siyi Chen, Yi Ma, and Qing Qu. Diffusion models learn low-dimensional distributions via subspace clustering, 2024b. URL https://arxiv.org/abs/2409.02426v1.

Zhangyang Wang, Peihao Wang, and Kai Wang. Position: Weight space should be a first-class generative AI modality. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=vu0ousWLde.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

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*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Zeng_Generative_Modeling_of_Weights_Generalization_or_Memorization_CVPR_2026_paper.pdf.

Baoquan Zhang, Chuyao Luo, Demin Yu, Xutao Li, Huiwei Lin, Yunming Ye, and Bowen Zhang. MetaDiff: Meta-learning with conditional diffusion for few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16687–16695, 2024a. URL https://ojs.aaai.org/index.php/AAAI/article/view/29608.

Huijie Zhang, Jinfan Zhou, Yifu Lu, Minzhe Guo, Peng Wang, Liyue Shen, and Qing Qu. The emergence of reproducibility and consistency in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60558–60590. PMLR, 2024b. URL https://proceedings.mlr.press/v235/zhang24cn.html.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4e9d8aeeab6120c3c83ccf95d4c211d3-Abstract-Conference.html.

A Detailed Experimental Setup

This appendix expands the common setup used by Protocol II and all WINS experiments in the main paper. The terminology and numerical results follow the main paper. In particular, the generator is called WINS, the diagnostic framework is called WeightAUDIT, and the five metrics are WDG, BDG, IX, UG, and PD in that order. A checkpoint denotes a complete evaluable parameter instance, θ denotes its weight vector, and f_θ denotes the network instantiated from that checkpoint.

A.1 Target Architecture

The controlled setting generates a complete TinyCNN checkpoint with no batch normalization or dropout. Inputs have shape $32 \times 32 \times 3$. For channel widths (c_1, c_2), both convolutions use a 3×3 kernel, unit stride, unit padding, and a bias. Each is followed by ReLU and 2×2 max pooling. The network is

$$\begin{aligned} h_1 &= \mathrm{Pool}_2\big(\mathrm{ReLU}\left(\mathrm{Conv}_{3\times3}^{3\to c_1}(x)\right)\big), \\ h_2 &= \mathrm{Pool}_2\big(\mathrm{ReLU}\left(\mathrm{Conv}_{3\times3}^{c_1\to c_2}(h_1)\right)\big), \\ f(x) &= \mathrm{Linear}_{64c_2\to10}\left(\mathrm{Flatten}(h_2)\right), \end{aligned}$$

where the final linear layer also has a bias. TinyCNN$(16, 8)$ and TinyCNN$(32, 32)$ contain $D =$ 6,738 and $D =$ 30,634 trainable parameters. TinyCNN3$(32, 64, 128)$ adds a third convolution, ReLU, and pooling block before the classifier and has $D =$ 113,738. Every checkpoint contains all weights and biases and is obtained at the end of an independent training run.

A.2 Datasets and Image-Level Protocol

Tables 6 and 7 record the image-level and checkpoint-training protocols for MNIST (LeCun et al., 1998), EMNIST (Cohen et al., 2017), KMNIST (Clanuwat et al., 2018), SVHN (Netzer et al., 2011), and CIFAR10 (Krizhevsky, 2009). EMNIST denotes the digits split. Whenever a class-balanced subset is requested, a fixed permutation with seed zero is followed by class-wise truncation. The image split and the checkpoint split are different objects: "held-out checkpoint" never means a held-out image.

Table 6: Image sources and overlap. WeightAUDIT images never overlap the checkpoint-training images because they come from the official test split. They do overlap the checkpoint evaluation set: the 2,000 images are either that set itself (MNIST) or a deterministic subset of it.

Dataset	Checkpoint-training images	Checkpoint evaluation images	WeightAUDIT images	Overlap with checkpoint evaluation
MNIST	Official train; 300/class (3,000)	Official test; 200/class (2,000)	Same fixed official-test subset; 200/class (2,000)	Exact same subset
EMNIST	Official train; 5,000/class (50,000)	Official test; 1,000/class (10,000)	First 200/class under the same fixed ordering (2,000)	Subset
KMNIST	Official train; 5,000/class (50,000)	Full official test (10,000)	First 200/class under the same fixed ordering (2,000)	Subset
CIFAR10	Full official train (50,000)	Full official test (10,000)	First 200/class under the same fixed ordering (2,000)	Subset
SVHN	Official train only, excluding the additional split; cap 5,000/class (49,607)	Official test; 1,000/class (10,000)	First 200/class under the same fixed ordering (2,000)	Subset

Following the main paper, all target networks are trained with Adam (Kingma & Ba, 2015) at an initial learning rate of 10^{-3} for 80 epochs. Their final checkpoints are retained without early stopping or validation-based model selection.

A.3 Checkpoint Pool Construction

For every dataset and target architecture, we independently train 3,256 target networks and retain their final checkpoints. A deterministic permutation with seed zero is applied once and defines the

Table 7: Dataset-specific preprocessing and augmentation. Augmentation is applied only while training target networks. WeightAUDIT evaluation is deterministic and unaugmented.

Dataset	Resize and channels	Numeric normalization	Checkpoint-training augmentation
MNIST	$28\times28 \rightarrow 32\times32$, bilinear; one channel repeated to three	$[0, 1]$; no mean–variance normalization	None in the TinyCNN$(16, 8)$ training protocol
EMNIST	$28\times28 \rightarrow 32\times32$, bilinear; one channel repeated to three; the native stored orientation is retained	$[0, 1]$; no mean–variance normalization	Zero-pad 4, random 32×32 crop, horizontal flip $p = 0.5$
KMNIST	$28\times28 \rightarrow 32\times32$, bilinear; one channel repeated to three	$[0, 1]$; no mean–variance normalization	Zero-pad 4, random 32×32 crop, horizontal flip $p = 0.5$
CIFAR10	Native 32×32 RGB; no resize	Divide uint8 values by 255; no mean–variance normalization	Zero-pad 4, random 32×32 crop, horizontal flip $p = 0.5$
SVHN	Native 32×32 RGB; no resize	Divide uint8 values by 255; no mean–variance normalization	$D = 6{,}738$: pad/crop and flip $p = 0.5$; $D = 30{,}634/113{,}738$: pad/crop, no flip

common split

$$N_{\text{total}} = 3{,}256, \qquad N_{\text{tr}} = 3{,}000, \qquad N_{\text{ho}} = 256.$$

The first 256 permuted indices form the held-out checkpoint reference and the remaining 3,000 form the generator-training pool. The two index sets are disjoint and exhaust the 3,256 checkpoints. Throughout this supplement, the total checkpoint pool contains 3,256 checkpoints. The term "training pool" and the symbol N in the checkpoint pool size sweep refer only to the checkpoints available for generator training and exclude the fixed held-out set. One member of this training pool defines the permutation-alignment reference. Held-out checkpoints never enter generator fitting.

The training-checkpoint mean accuracies reported in the main paper are 0.934 for MNIST, 0.951 for EMNIST, 0.728 for KMNIST, 0.685 for SVHN, and 0.603 for CIFAR10. They are fixed-image evaluation summaries of the corresponding 3,000-checkpoint generator-training split.

A.4 Information Isolation and Reported Settings

The 256 held-out checkpoints and the fixed 2,000 WeightAUDIT images never contribute to target network training or a generator-training update. They are used only after a generator configuration has finished training, for WeightAUDIT measurement. A β-sweep is not part of a single generator training run: every point in the sweep is trained independently on the same 3,000-checkpoint training split and then evaluated afterward.

The default KMNIST TinyCNN$(16, 8)$ operating point uses $\beta = 0.5$, while the reported MNIST and EMNIST operating points both use $\beta = 3$. The β values used for the six CIFAR10/SVHN architecture configurations are recorded in Table 8.

A.5 Generated Populations and Statistical Unit

Every reported generator configuration uses five independently trained generator seeds. Each seed produces 256 generated checkpoints without fine-tuning or accuracy-based filtering. Reported values are means over the five seeds, and the generator seed is the unit of uncertainty. Profile figures show 95% confidence intervals at the generator-seed level.

B Complete WeightAUDIT Profiles

The main paper presents the most representative quantities for the behavioral-supervision and checkpoint-pool sweeps. This appendix visualizes the complete trajectories of the mean accuracy of generated checkpoints and all five WeightAUDIT metrics. WDG, BDG, and IX are normalized against held-out checkpoints. PD is normalized against the generator-training checkpoint population. UG is normalized against training-checkpoint accuracy and a behavior-matched noise baseline.

Table 8: Target configurations and β values used for the CIFAR10 and SVHN architecture study.

Target CNN	D	SVHN		CIFAR10	
		Checkpoint augmentation	β	Checkpoint augmentation	β
TinyCNN$(16, 8)$	6,738	Pad/crop, flip	0.3	Pad/crop, flip	0.5
TinyCNN$(32, 32)$	30,634	Pad/crop, no flip	0.6	Pad/crop, flip	0.5
TinyCNN3$(32, 64, 128)$	113,738	Pad/crop, no flip	0.3	Pad/crop, flip	0.7

B.1 Behavioral-Supervision Sweep

The upper scan in Fig. 5 complements the main-paper summary with all five metrics for $\beta \in [0, 2]$ on KMNIST. Across the supervised range, WDG remains between 0.94 and 0.98 and the near-replica rate remains zero. The shaded interval $\beta \in [0.6, 2.0]$ marks a sustained contraction regime in which PD stays below its training reference while accuracy and UG remain high. All settings use the same target architecture, 3,000-checkpoint generator-training pool, and 256 generated checkpoints per seed.

B.2 Checkpoint-Pool-Size Sweep

The lower scan in Fig. 5 reports the complete trajectories for the checkpoint-pool-size sweep. Here, N is the number of generator-training checkpoints available to WINS. The held-out population remains fixed and separate. WDG, BDG, and IX use that population, while PD uses all N training checkpoints. The unplotted near-replica rate is a supporting effect size rather than a WeightAUDIT metric.

Beyond the trends summarized in the main paper, the near-replica rate falls from 1.00 at the smallest pool to 0.01 at $N = 64$ and remains zero from $N = 128$ onward. At $N = 2048$, IX, UG, and PD are 1.34, 1.12, and 1.06.

A targeted follow-up would vary training duration at fixed checkpoint pools while tracking denoising loss, task loss, and WeightAUDIT profiles. Repeated pool draws with matched checkpoint accuracy would then vary parameter and behavioral diversity at fixed N. Comparing fixed update budgets with matched checkpoint exposure would help distinguish optimization effects from pool composition. If the dip weakens with longer training while training dependence remains low, this would support an optimization contribution to the dip.

C Larger CNN Target Configurations

Table 8 maps the three parameter scales to the six CIFAR10/SVHN configurations. Every target network is trained for 80 epochs. Each WINS configuration uses the 3,000/256 checkpoint split described above, five generator seeds, and 8,000 updates, and produces 256 generated checkpoints per seed.

WINS directly generates complete parameter vectors from $D = 6{,}738$ to $113{,}738$ without a VAE or a separate dimensionality-reducing latent bottleneck, and no parameter coordinate is dropped, preserving the full parameter representation and enabling direct WeightAUDIT analysis. Because memory use, token count, and training time increase with D, future work will investigate block-wise generation, hierarchical modeling, and compressed representations that can recover complete weights for larger models.

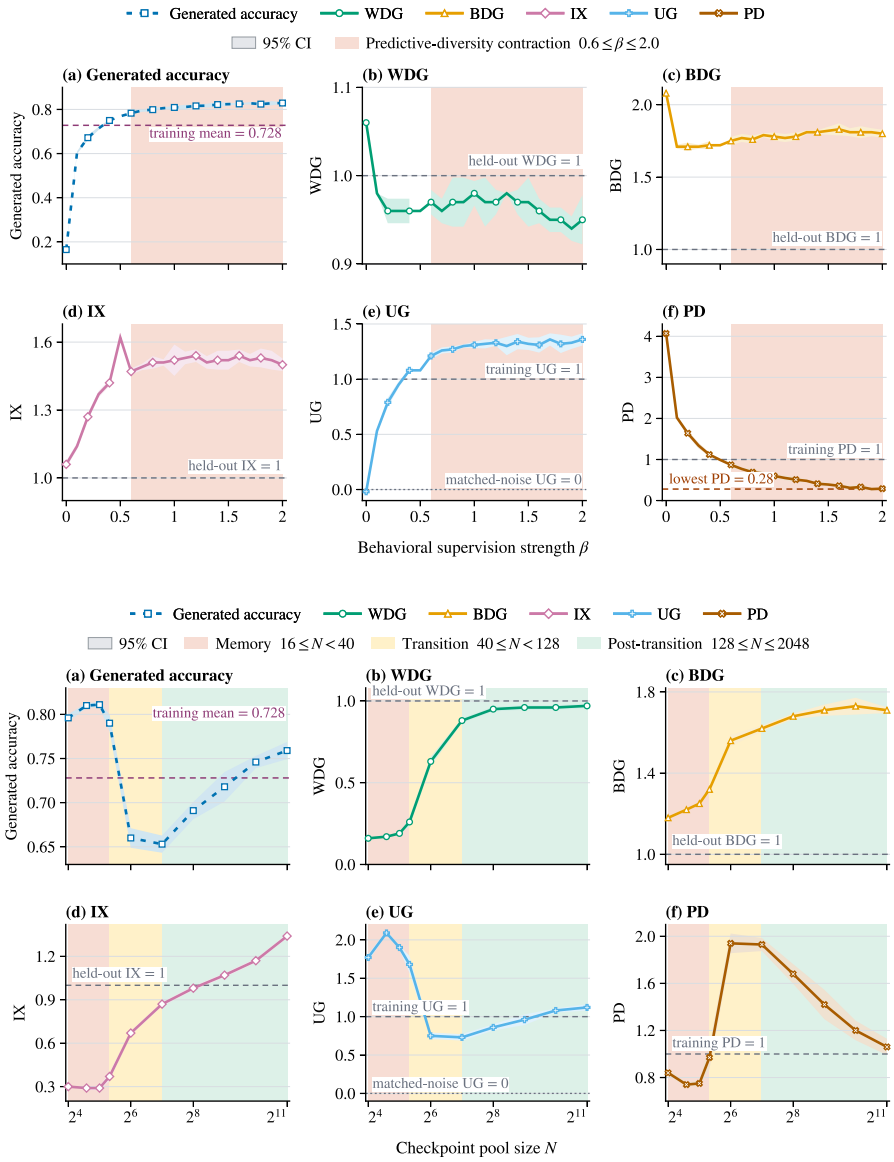


Figure 5: Complete WeightAUDIT scans on KMNIST. The upper scan varies behavioral supervision β, and the lower scan varies the generator-training pool size N. Curves show means over five generator seeds and shaded bands show 95% confidence intervals. In both scans, the accuracy reference is the training-checkpoint mean of 0.728. WDG, BDG, and IX use held-out checkpoint references, while PD uses the corresponding generator-training population. The lower scan excludes held-out checkpoints from N, and its background bands identify the three pool-size regimes.

D WeightAUDIT Implementation Details

D.1 Common Evaluation Procedure

All generated, training, and held-out checkpoints are instantiated in the target architecture and evaluated on the same fixed 2,000-image set. Weight-space measurements use permutation-aligned weights. Behavioral measurements use predictions and therefore do not require parameter alignment. The generated checkpoint set contains 256 checkpoints, matching the held-out checkpoint set in size. We write $G = |\Theta_{\mathrm{gen}}|$, $H = |\Theta_{\mathrm{ho}}|$, and $N_{\mathrm{tr}} = |\Theta_{\mathrm{tr}}|$.

A score near one means that the generated population is comparable to the metric-specific reference on that diagnostic axis. WDG, BDG, and IX use the held-out population, PD uses the training-checkpoint population, and UG uses training accuracy together with behavior-matched noise. For BDG, values above one indicate lower average maximum overlap with training error sets than the held-out reference. For IX, they indicate a larger mean normalized residual from the training affine span. For PD, they indicate greater mean pairwise predictive disagreement than the training population. These ratios compare specific statistics, not full distributions, and one is a reference level rather than an optimum. Table 2 illustrates why larger values do not imply better generation. RPG has higher BDG, IX, and PD than WINS (2.11, 2.17, and 4.20 versus 1.72, 1.62, and 0.99), but lower accuracy (0.117 versus 0.769) and UG (-0.06 versus 1.16). We therefore interpret the diagnostic profile jointly.

D.2 Weight Dependence Gap and Replica Diagnostic

Each query and every training checkpoint are first mapped to the shared permutation frame. For an aligned query q, its distance to the training pool is

$$d_{\mathrm{tr}}(q) = \min_{\theta^{(i)} \in \Theta_{\mathrm{tr}}} \|q - \theta^{(i)}\|_2.$$

We evaluate the same distance function for generated and held-out queries, using only Θ_{tr} as the nearest-neighbor database. The finite sample means are

$$\bar{d}_{\mathrm{gen}} = \frac{1}{G} \sum_{q \in \Theta_{\mathrm{gen}}} d_{\mathrm{tr}}(q),$$
$$\bar{d}_{\mathrm{ho}} = \frac{1}{H} \sum_{q \in \Theta_{\mathrm{ho}}} d_{\mathrm{tr}}(q).$$

The Weight Dependence Gap is

$$M_{\mathrm{WDG}} = \frac{\bar{d}_{\mathrm{gen}}}{\bar{d}_{\mathrm{ho}}}.$$

Thus, $M_{\mathrm{WDG}} \approx 1$ means that generated checkpoints are as far from the training pool as independently trained held-out checkpoints. Values below one indicate contraction toward the training pool. When $\bar{d}_{\mathrm{ho}} \leq \varepsilon_{\mathrm{WDG}}$, the denominator is treated as degenerate and WDG is unavailable.

WDG is a population-level ratio. We additionally report a near-replica effect size. The leave-one-out training separation is

$$\bar{d}_{\mathrm{tr,tr}} = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} \min_{j \neq i} \|\theta^{(i)} - \theta^{(j)}\|_2.$$

The empirical near-replica rate is

$$R_{\mathrm{copy}} = \frac{1}{G} \sum_{q \in \Theta_{\mathrm{gen}}} \mathbb{I}\left[d_{\mathrm{tr}}(q) < 0.3\, \bar{d}_{\mathrm{tr,tr}}\right].$$

The threshold is an interpretable effect-size control. The replica rate is not substituted for WDG in the five-dimensional profile. The WDG denominator is declared degenerate at $\varepsilon_{\mathrm{WDG}} = 10^{-12}$.

D.3 Behavior Dependence Gap

Let the fixed evaluation set be $\mathcal{D}_{\text{eval}} = \{(x_m, y_m)\}_{m=1}^{M}$ and let $\widehat{y}_\theta(x) = \arg\max_c f_\theta(x)_c$. Identifying each evaluation pair with its index m, we write the error set of a network as

$$E(\theta) = \{m \in \{1, \dots, M\} : \widehat{y}_\theta(x_m) \neq y_m\}.$$

For two networks, the error-set similarity is

$$J(\theta, \phi) = \begin{cases} \dfrac{|E(\theta) \cap E(\phi)|}{|E(\theta) \cup E(\phi)|}, & |E(\theta) \cup E(\phi)| > 0, \\ 1, & E(\theta) = E(\phi) = \varnothing. \end{cases}$$

The empty-union branch treats two error-free predictors as behaviorally identical. For any generated or held-out query q, its maximum behavioral similarity to the training pool is

$$s_{\text{tr}}(q) = \max_{\theta^{(i)} \in \Theta_{\text{tr}}} J(q, \theta^{(i)}).$$

This implements the similarity $s(\theta)$ defined in the main paper on the finite checkpoint sets, with q denoting the parameter vector of a generated or held-out checkpoint. We then compute the mean behavioral dissimilarities

$$b_{\text{gen}} = 1 - \frac{1}{G} \sum_{q \in \Theta_{\text{gen}}} s_{\text{tr}}(q),$$

$$b_{\text{ho}} = 1 - \frac{1}{H} \sum_{q \in \Theta_{\text{ho}}} s_{\text{tr}}(q).$$

The Behavior Dependence Gap is

$$M_{\text{BDG}} = \frac{b_{\text{gen}}}{b_{\text{ho}}}.$$

A value near one means that generated error patterns are separated from the training pool at the held-out reference level. Values below one indicate replication of training error patterns even when parameter vectors are not Euclidean near-neighbors. If $b_{\text{ho}} \leq \varepsilon_{\text{BDG}}$, BDG is reported as unavailable. We use the empty-union branch above, resolve exact logit ties by the smallest class index, and set $\varepsilon_{\text{BDG}} = 10^{-12}$.

D.4 Interpolation Exclusion

Following the main paper, IX is computed from raw, permutation-aligned weights rather than WINS's standardized coordinates. For a query q, the normalized off-span residual is

$$r_\perp(q) = \frac{\|q - \Pi_{\mathcal{S}_{\text{tr}}}(q)\|_2}{\max\{\|q - \mu_{\text{tr}}\|_2, \varepsilon_{\text{norm}}\}}.$$

$$M_{\text{IX}} = \frac{\frac{1}{G} \sum_{q \in \Theta_{\text{gen}}} r_\perp(q)}{\frac{1}{H} \sum_{q \in \Theta_{\text{ho}}} r_\perp(q)},$$

where $\varepsilon_{\text{norm}} = 10^{-12}$.

IX is unavailable when the held-out mean residual is below 0.02. For a method with a lossy reconstruction path g, we also require

$$\bar{r}_{\text{rec}} = \frac{1}{N_{\text{tr}}} \sum_{\theta \in \Theta_{\text{tr}}} r_\perp(g(\theta)) \leq 0.5, \qquad \bar{r}_{\text{rec}} < 0.8\, \bar{r}_{\text{gen}}.$$

If these conditions fail, or if the generated, held-out, and Gaussian-control residual means all exceed 0.98, IX is reported as unavailable.

Supplementary in-span diagnostic. For calibration populations confined to the training affine span, we also report their normalized in-span spread:

$$M_{\text{IX}}^{\parallel} = \frac{\frac{1}{G} \sum_{q \in \Theta_{\text{gen}}} \|\Pi_{\mathcal{S}_{\text{tr}}}(q) - \mu_{\text{tr}}\|_2}{\frac{1}{H} \sum_{q \in \Theta_{\text{ho}}} \|\Pi_{\mathcal{S}_{\text{tr}}}(q) - \mu_{\text{tr}}\|_2}.$$

This quantity is used only in the calibration analysis and never replaces M_{IX}.

D.5 Utility Gain and Behavior-Matched Noise

UG asks whether separation from the training pool preserves task utility. For any checkpoint θ, define its fixed-set accuracy as

$$a(\theta) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\widehat{y}_\theta(x_m) = y_m].$$

The behavioral separation of a query is

$$b_{\mathrm{tr}}(q) = 1 - s_{\mathrm{tr}}(q),$$

so the generated population has mean separation

$$\bar{b}_{\mathrm{gen}} = \frac{1}{G} \sum_{q \in \Theta_{\mathrm{gen}}} b_{\mathrm{tr}}(q) = b_{\mathrm{gen}}.$$

We construct a Gaussian-noise frontier in the same aligned and standardized parameter space used to train WINS. Let $z(\theta)$ denote the standardized coordinate of an aligned training checkpoint and let $\mathcal{S} = (0.2, 0.4, 0.6, 0.8, 1.0, 1.3, 1.6, 2.0)$ be the evaluated noise grid. One hundred anchors are sampled without replacement from the training pool using fixed seed zero. A fresh Gaussian draw is then made at each scale. At scale σ_k, select training anchors $\theta^{(a_\ell)}$ and form

$$\begin{aligned} z(q_{k\ell}) &= z(\theta^{(a_\ell)}) + \sigma_k \epsilon_{k\ell}, \\ \epsilon_{k\ell} &\sim \mathcal{N}(0, I_D), \qquad \ell = 1, \ldots, L_k. \end{aligned}$$

Each $q_{k\ell}$ is inverse-standardized, instantiated in the target architecture, and evaluated on $\mathcal{D}_{\mathrm{eval}}$. The finite-sample frontier coordinates are

$$\bar{b}_k = \frac{1}{L_k} \sum_{\ell=1}^{L_k} b_{\mathrm{tr}}(q_{k\ell}),$$

$$A_k = \frac{1}{L_k} \sum_{\ell=1}^{L_k} a(q_{k\ell}).$$

The matched grid point is

$$k^\star \in \arg\min_{1 \le k \le K} \left| \bar{b}_k - \bar{b}_{\mathrm{gen}} \right|, \qquad A_{\mathrm{noise}} = A_{k^\star}.$$

Thus the baseline has, among the evaluated Gaussian populations, the behavioral separation closest to the generated population. The population accuracies used in the numerator and denominator are

$$A_{\mathrm{gen}} = \frac{1}{G} \sum_{q \in \Theta_{\mathrm{gen}}} a(q),$$

$$A_{\mathrm{tr}} = \frac{1}{N_{\mathrm{tr}}} \sum_{\theta \in \Theta_{\mathrm{tr}}} a(\theta).$$

Utility Gain is

$$M_{\mathrm{UG}} = \frac{A_{\mathrm{gen}} - A_{\mathrm{noise}}}{A_{\mathrm{tr}} - A_{\mathrm{noise}}}.$$

A value of one means that the mean accuracy of generated checkpoints reaches the training-checkpoint mean relative to the matched noise baseline. A value of zero means that it only reaches the matched-noise accuracy. Negative values indicate accuracy below that baseline, while values above one indicate accuracy above the training-checkpoint mean. UG is not clipped. If $|A_{\mathrm{tr}} - A_{\mathrm{noise}}| \le \varepsilon_{\mathrm{UG}}$, the normalization is degenerate and UG is unavailable. The averaged-checkpoint and Gaussian-noise controls are not substituted for this noise frontier. The selected point is the first minimum of the absolute separation difference; the same rule consequently selects the nearest endpoint when the generated separation lies outside the grid range. No interpolation between grid points is used, and $\varepsilon_{\mathrm{UG}} = 10^{-12}$.

D.6 PREDICTIVE DIVERSITY

PD uses the generator-training checkpoints as its reference. For two checkpoints, define fixed-set predictive disagreement as

$$\delta(\theta^{(i)}, \theta^{(j)}) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\widehat{y}_{\theta^{(i)}}(x_m) \neq \widehat{y}_{\theta^{(j)}}(x_m)].$$

For a population $\Theta = \{\theta^{(i)}\}_{i=1}^{n}$ with $n \geq 2$, its empirical predictive diversity is the unordered, off-diagonal pair average

$$\mathrm{Div}(\Theta) = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \delta(\theta^{(i)}, \theta^{(j)}).$$

Predictive Diversity uses the complete generator-training population as its reference:

$$M_{\mathrm{PD}} = \frac{\mathrm{Div}(\Theta_{\mathrm{gen}})}{\mathrm{Div}(\Theta_{\mathrm{tr}})}.$$

A value near one means that the generated population has training-reference predictive diversity. Values below one indicate contraction or predictive collapse. Values above one indicate a more dispersed predictor population, but do not by themselves imply better generation. PD is not clipped and does not require parameter alignment. If either population has fewer than two members, or if $\mathrm{Div}(\Theta_{\mathrm{tr}}) \leq \varepsilon_{\mathrm{PD}}$, PD is unavailable. All N_{tr} training checkpoints are included and $\varepsilon_{\mathrm{PD}} = 10^{-12}$.

D.7 GROUND-TRUTH CALIBRATION CONTROLS AND CONTROLLED FAILURE CASES

Control protocol. We construct ground-truth populations for which the relevant failure is known before applying WeightAUDIT. To keep this check independent of generator training and hyperparameter selection, we use one fixed CIFAR10 TinyCNN$(32, 32)$ calibration pool with $D = 30{,}634$ parameters. This is a deterministic metric calibration rather than a generator experiment in the common-pool comparison. We use the same fixed 2,000-image evaluation set, alignment, distance convention, IX gates, 100-member UG noise frontier, and complete training-pool PD reference. No generator is trained for these controls. The populations are deterministic, and the standard deviations in Table 9 describe variation across population members rather than repeated-run uncertainty.

Control construction. The healthy population is Θ_{ho}. A fixed 256-member subset $\Theta_{\mathrm{tr}}^{\mathrm{ctrl}}$ constructs the copy controls but is not the PD reference. Parameter copies are its distinct members. Affine controls average 16 aligned training checkpoints with equal coefficients and seed 17. The UG anchors use the $\sigma = 0.8$ and $\sigma = 2$ points of the fixed noise frontier, while the collapse control repeats the best held-out checkpoint 256 times. For behavior memory, seed 23 rescales each ReLU channel of every member of $\Theta_{\mathrm{tr}}^{\mathrm{ctrl}}$ by a factor in $\{1/2, 1, 2\}$ and inversely rescales the next layer. This preserves the network function while changing its parameters.

Expected and observed signatures. Table 9 recovers the intended signatures. Exact copies have WDG, BDG, and IX near zero, while $M_{\mathrm{IX}}^{\parallel} = 1.68$ describes their in-span spread. ReLU rescaling preserves predictions but gives WDG $= 1.15$ and BDG $= 0$, separating parameter from behavior memory. Affine averages have IX below 0.01, and $M_{\mathrm{IX}}^{\parallel} = 0.42$ quantifies contraction within the span. The noise and repeated-checkpoint controls further show that high WDG or PD does not imply utility and that a healthy score on other axes does not prevent predictive collapse.

Behavior-memory case with a healthy comparator. Figure 6 makes behavioral memory visible at both image and population levels. The training source and its closest-accuracy held-out comparator both have 71.6% accuracy and 568 errors. Their prediction agreement is 0.8225 and error-set Jaccard is 0.6205. The function-preserving copy changes the parameters but raises both similarities to one.

Parameter, affine, utility, and diversity cases. Figure 7 supplies a common-scale counterpart for parameter memorization, interpolation, ineffective generation, and predictive collapse.

Table 9: Ground-truth calibration of all five WeightAUDIT metrics on the CIFAR10 TinyCNN$(32, 32)$ pool. Accuracy is mean $\pm$ standard deviation across population members, shown in percentage points. The IX column reports M_{IX}. For exact copies and affine averages, $M_{\mathrm{IX}}^{\|}$ is 1.68 and 0.42, respectively. PD uses the complete training pool. Following the main paper, light gray marks $0.40 \leq M < 0.70$ and dark gray marks $M < 0.40$.

Ground-truth population	Known truth	Members	Accuracy (%)	WeightAUDIT (healthy reference = 1)				
				WDG	BDG	IX	UG	PD
Independent held-out checkpoints	Healthy reference	256	72.66±0.69	1.00	1.00	1.00	1.00	1.00
Exact training-checkpoint copies	Parameter memory	256	72.59±0.65	< 0.01	0.00	< 0.01	1.00	1.00
Function-preserving ReLU rescalings	Behavior memory	256	72.59±0.65	1.15	0.00	0.96	1.00	1.00
Affine averages of 16 checkpoints	Interpolation	256	33.49±5.16	0.64	2.00	< 0.01	0.21	3.38
Behavior-matched Gaussian noise, $\sigma = 0.8$	UG zero anchor	100	14.08±3.70	0.64	2.12	0.76	0.00	4.87
One best held-out checkpoint repeated	Predictive collapse	256	74.70±0.00	0.99	1.03	1.01	1.07	0.00
Nonfunctional high noise, $\sigma = 2$	Novelty $\neq$ utility	100	10.42±1.54	1.59	2.13	1.09	0.00	4.91

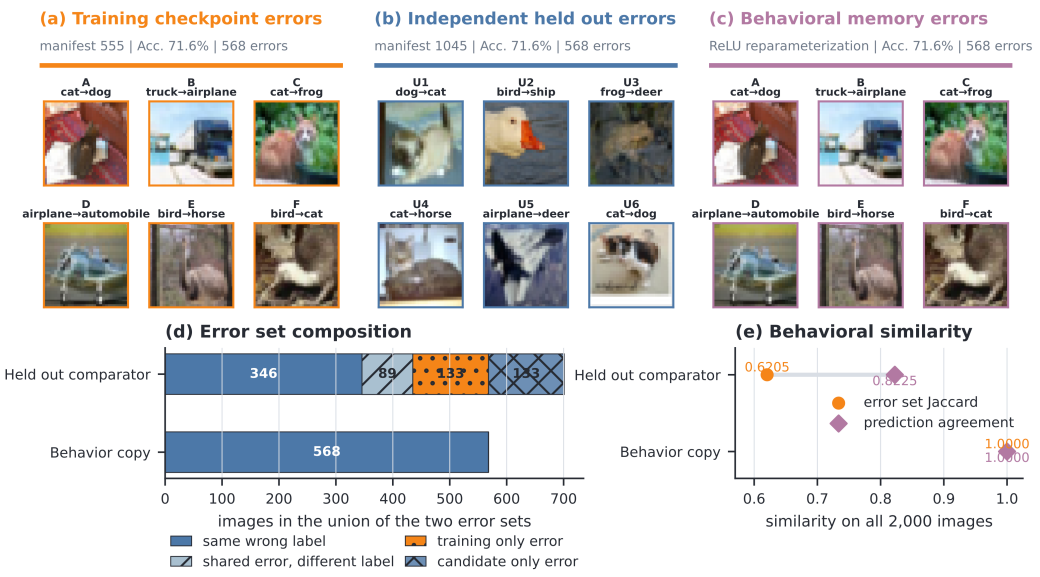


Figure 6: Behavior-memory comparison between a training source, an equally accurate held-out model, and a function-preserving copy. Panels (a)–(c) show representative errors, while panels (d) and (e) summarize error overlap and whole-set similarity. The held-out model has distinct mistakes, whereas the behavior copy exactly reproduces the source predictions.

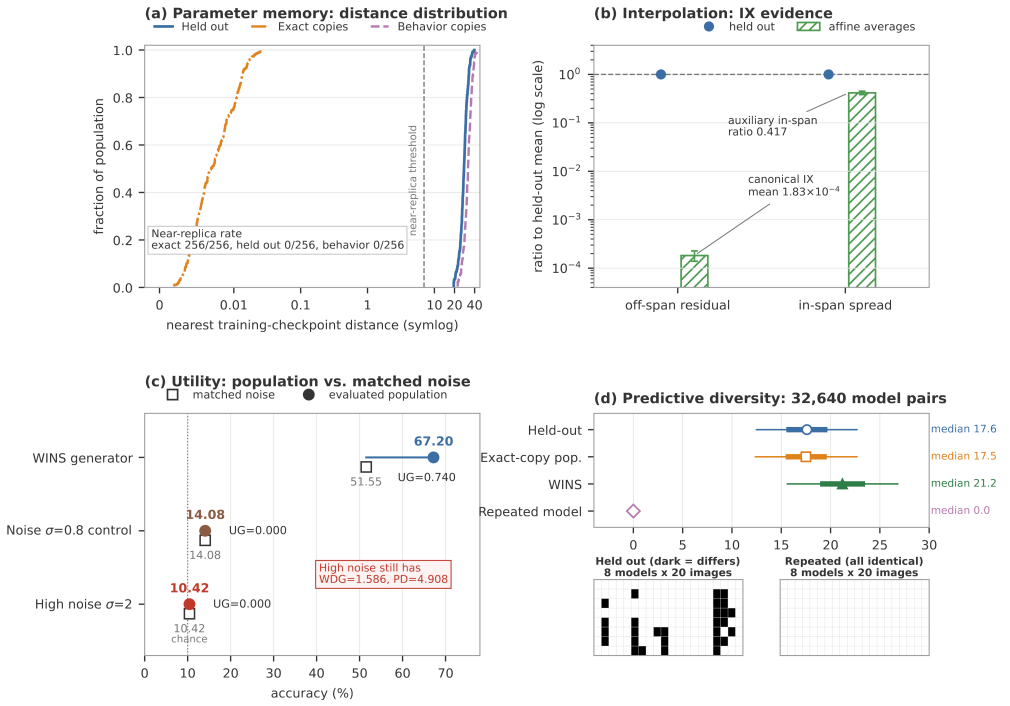


Figure 7: Controlled comparisons for the other failure modes. (a) Empirical cumulative distributions of nearest-training distance. Exact copies are near replicas; behavior-preserving copies are not, which separates parameter memory from behavior memory. (b) Held-out and affine distributions for the IX residual and the in-span diagnostic. Affine averages have IX below 0.01 and $M_{\mathrm{IX}}^{\parallel} = 0.417$. (c) Paired accuracy comparisons against each population's behavior-matched Gaussian baseline. WINS lies above its baseline, but the two noise controls coincide with theirs; $\sigma = 2$ remains unhelpful despite WDG $= 1.586$ and PD $= 4.908$. (d) The 5th–95th percentile, interquartile range, and median of all 32,640 pairwise disagreements. Held-out checkpoints, exact-copy populations, and WINS remain diverse, while repeated checkpoints have zero disagreement. The prediction tiles compare the first eight models on the first 20 evaluation images; dark cells mark disagreement with the first model. In panels (c) and (d), WINS denotes the main-paper CIFAR10 TinyCNN$(32, 32)$ configuration.

D.8 Metric Bounds and Threshold Interpretation

Bounds under fixed references. The normalized metrics are not restricted to $[0, 1]$. Let

$$b_{\mathrm{ho}} = 1 - \mathbb{E}_{\theta\sim\Theta_{\mathrm{ho}}}[s(\theta)], \qquad \bar{r}_{\mathrm{ho}} = \mathbb{E}_{\theta\sim\Theta_{\mathrm{ho}}}[r_{\perp}(\theta)], \qquad v_{\mathrm{tr}} = \mathrm{Div}(\Theta_{\mathrm{tr}}).$$

For valid positive reference denominators, the metrics satisfy

$$0 \le M_{\mathrm{WDG}} < \infty,$$
$$0 \le M_{\mathrm{BDG}} \le b_{\mathrm{ho}}^{-1},$$
$$0 \le M_{\mathrm{IX}} \le \bar{r}_{\mathrm{ho}}^{-1},$$
$$0 \le M_{\mathrm{PD}} \le v_{\mathrm{tr}}^{-1}.$$

These bounds follow from nonnegative parameter distances, $s(\theta) \in [0, 1]$, $r_{\perp}(\theta) \in [0, 1]$, and predictive disagreement rates in $[0, 1]$. In particular, the training mean belongs to the training affine span, so the distance to that span cannot exceed the distance to the training mean. The bounds depend on the reference statistics and need not be attainable for a given architecture and evaluation set.

Zero and reference values. WDG equals zero when every generated checkpoint coincides with a training checkpoint in the aligned parameter space. BDG equals zero when each generated check-

point has the same evaluation error set as some training checkpoint. IX equals zero when all generated checkpoints lie in the training affine span. PD equals zero when all generated checkpoints make identical predictions on the evaluation set. For these four metrics, one denotes equality with the corresponding reference statistic. Values above one denote larger measured separation, residuals, or disagreement, rather than a universal improvement in generation quality.

Utility Gain. Let $\Delta = A_{\mathrm{tr}} - A_{\mathrm{noise}}$. For fixed reference accuracies in $[0, 1]$ and $\Delta \neq 0$, the bounds implied by $A_{\mathrm{gen}} \in [0, 1]$ are

$$M_{\mathrm{UG}} \in \begin{cases} \left[-\dfrac{A_{\mathrm{noise}}}{\Delta}, \dfrac{1 - A_{\mathrm{noise}}}{\Delta}\right], & \Delta > 0, \\ \left[\dfrac{1 - A_{\mathrm{noise}}}{\Delta}, -\dfrac{A_{\mathrm{noise}}}{\Delta}\right], & \Delta < 0. \end{cases}$$

UG equals zero at the matched noise accuracy and one at the training mean accuracy. When $\Delta > 0$, negative values indicate accuracy below the matched noise baseline, values between zero and one indicate accuracy between the noise and training references, and values above one indicate accuracy above the training mean. Values near zero therefore indicate little normalized gain over the matched noise baseline. When $\Delta < 0$, UG decreases as generated accuracy increases, so the directional interpretations reverse. UG is not clipped.

Numerical and applicability thresholds. The implementation uses $\epsilon = 10^{-12}$ for the WDG, BDG, and PD reference denominators, the absolute UG denominator $|\Delta|$, and the norm safeguard in $r_{\perp}$. WDG, BDG, and PD are unavailable when their reference denominator is at most ϵ, and UG is unavailable when $|\Delta| \leq \epsilon$. PD also requires at least two checkpoints in each population.

The IX rules additionally require $\bar{r}_{\mathrm{ho}} \geq 0.02$. For a method with a lossy reconstruction map g, the mean reconstruction residual must satisfy $\bar{r}_{\mathrm{rec}} \leq 0.5$ and $\bar{r}_{\mathrm{rec}} < 0.8\bar{r}_{\mathrm{gen}}$. IX is also unavailable when the generated, held-out, and Gaussian-control mean residuals all exceed 0.98. These conditions determine whether the diagnostic is reported, rather than defining generation quality.

Display cutoffs and the replica diagnostic. The table shading uses light gray for $0.40 \leq M < 0.70$ and dark gray for $M < 0.40$. These cutoffs provide a common visual convention for reduced scores relative to the reference level. They are descriptive flags, not statistical significance thresholds or universal boundaries between successful and failed generation. An unshaded score does not by itself establish functional quality.

The separate near-replica diagnostic uses $d_{\mathrm{tr}}(q) < 0.3\bar{d}_{\mathrm{tr,tr}}$, as defined in Appendix D.2. This distance cutoff is distinct from the table shading cutoffs and does not replace WDG.

E WINS Implementation Details

E.1 Permutation Alignment

WINS uses Git Re-Basin (Ainsworth et al., 2023) to align every training checkpoint to one common training-pool reference. Coordinate descent runs for eight iterations and permutes convolutional channels together with their corresponding inputs in the next layer. After alignment, parameter coordinate j of checkpoint $\tilde{\theta}^{(i)}$ is standardized over the 3,000 training checkpoints as

$$z_j^{(i)} = \frac{\tilde{\theta}_j^{(i)} - \mu_j}{\sigma_j}, \qquad \sigma_j = \mathrm{std}_i\left(\tilde{\theta}_j^{(i)}\right) + 10^{-6}.$$

No dimension is dropped. The affine transformation is inverted after denoising.

E.2 Tokenization and Transformer Denoiser

The D-dimensional standardized weight vector is divided into contiguous blocks of 128 values and the last block is zero-padded. Each token has an independent $128 \to 256$ linear encoder and $256 \to 128$ linear decoder. The encoded sequence receives a learned positional embedding. A 128-dimensional sinusoidal time representation is transformed by a $128 \to 256 \to 256$ SiLU MLP and

Table 10: WINS token counts, padding, denoiser parameter counts, and diffusion horizons for the three target sizes.

D	Tokens	Padded D	Pad	Denoiser params.	T
6,738	53	6,784	46	8,344,704	400
30,634	240	30,720	86	20,719,616	400
113,738	889	113,792	54	63,667,840	1,000

added to every token. The backbone contains six Transformer encoder layers (Vaswani et al., 2017), width 256, eight attention heads, FFN width 1,024, GELU, and zero dropout.

E.3 Diffusion and Optimization

The reported common-pool WINS configuration is a DDPM (Ho et al., 2020) with a cosine cumulative noise schedule (Nichol & Dhariwal, 2021) using $s = 0.008$ and the horizon in Table 10. At each update, a checkpoint index $I \sim \mathrm{Uniform}\{1, \ldots, N\}$ is sampled and the clean state is set to $z_0 = z^{(I)}$. Independently, $t \sim \mathrm{Uniform}\{0, \ldots, T-1\}$. The implementation labels the T diffusion steps from zero to $T-1$. Thus, implementation label $t = T-1$ corresponds to diffusion step T, and hence to the noisiest state z_T, in the main paper. The diffusion minibatch size is 64 and the task minibatch size is 256. The denoiser predicts the clean standardized state as $\hat{z}_0$, and the reconstruction MSE is computed between $\hat{z}_0$ and z_0. For $\beta > 0$, parameter recovery converts $\hat{z}_0$ into the complete network parameters used for behavioral cross-entropy.

The optimizer is AdamW (Loshchilov & Hutter, 2019) with learning rate 3×10^{-4}, $\mathrm{betas} = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 0.01. All configurations use 8,000 updates, 1,000 linear-warmup updates, cosine decay to 5% of the initial learning rate, gradient clipping at 1.0, and EMA decay 0.999. The terminal EMA weights are used for sampling; no intermediate checkpoint is selected by validation accuracy.

E.4 Differentiable Behavioral Supervision

At each iteration, parameter recovery converts $\hat{z}_0$ into complete parameters, and WINS evaluates cross-entropy on a 256-image task minibatch through the recovered network. The objective adds this term with weight β to the dimension-normalized reconstruction loss between $\hat{z}_0$ and z_0. Generation uses no refinement at test time, rejection sampling, or selection by accuracy.

E.5 Sampling Procedure

Sampling uses the standard ancestral DDPM posterior (Ho et al., 2020) for a z_0-predicting denoiser. At the final transition, the returned prediction is denoted by $z_0^{\mathrm{gen}} := \hat{z}_0$, which parameter recovery converts into θ^{gen}. For the sampling-efficiency study, we select K approximately uniform time labels from $T-1$ to zero, round them to the nearest integer, and remove duplicates. Consecutive labels $t > t'$ use the effective transition

$$\alpha_{\mathrm{eff}} = \frac{\bar{\alpha}_t}{\bar{\alpha}_{t'}}.$$

The same ancestral posterior is then applied across the selected labels. Thus, $K = 1$ performs one z_0 prediction at $t = T-1$, while $K = T$ recovers the complete ancestral chain.

F Evaluation Protocol Details

F.1 Protocol I: Original-Setting Audit

Following the main paper, Protocol I audits each method in its own optimized setting while preserving its original task, architecture, and checkpoint pool. Table 12 records the target setting only. Every method uses five generator seeds and produces 256 generated checkpoints per seed.

Table 11: Protocol-II method adaptations and training settings under the common KMNIST configuration.

Method	Adaptation relative to the original method	Optimizer, budget, and selection
WINS	Full aligned checkpoint; differentiable behavioral supervision	WINS configuration above; terminal EMA and the reported fixed β
p-diff (2024a)	Full six-tensor checkpoint rather than the native BN subset; 64-value slices	AdamW; VAE 2,000 plus DDPM 8,000 steps; batch 50; $2 \times 10^{-5}/10^{-4}$ learning rates, weight decay 10^{-5}, cosine schedule; terminal checkpoint
Hyper-Representations (2022a)	6,738-dimensional TinyCNN representation with native top-30% KDE sampling	Adam, 10^{-4}, weight decay 10^{-9}, 350 epochs, batch 500, plateau-based learning-rate schedule; 25,000 permutation augmentations; terminal epoch then top-30% KDE with bandwidth 0.002
Tina (2024)	New parameter splits; constant 10-kana CLIP (Radford et al., 2021) condition; classifier augmentation retained	AdamW, 4×10^{-4}, weight decay 0.1, batch 64, 300 epochs, 500-step warmup plus cosine, clip 0.1, EMA 0.9999; terminal EMA
RPG (2025)	KMNIST TinyCNN target with full-vector tokens; remaining small-CNN settings retained	AdamW, 3×10^{-5}, weight decay 0, batch 6, 5,000 steps, cosine schedule; terminal checkpoint with no separate validation rule
D2NWG (2025)	6,738-dimensional weight VAE; constant CLIP image condition (10 classes $\times$ 5 images $\times$ 512)	Stage 1 Adam, 1.115×10^{-3}, betas (0.5, 0.9), 3,000 epochs; stage 2 AdamW, 1.133×10^{-4}, 1,000 epochs, batch 64, 1,024 DDPM steps, 10,000-step warmup; terminal checkpoints
HyperDiffusion (2023)	Six TinyCNN parameter blocks / direct full weights	AdamW, 2×10^{-4}, batch 32, 6,000 epochs, $T = 500$, learning rate multiplied by 0.9 every 200 epochs; terminal-epoch DDIM (Song et al., 2021a)
DeepWeight Flow (2026)	Full aligned vectors with permutation alignment against the first training checkpoint as reference	AdamW, 5×10^{-4}, weight decay 10^{-5}, betas (0.9, 0.95), batch 8, 30,000 iterations, cosine schedule, 100-step fourth-order Runge–Kutta integration; terminal iterate

Table 12: Protocol-I target settings. Every method is evaluated with five generator seeds and 256 generated checkpoints per seed.

Method	Original target setting
HyperDiffusion (2023)	ShapeNet airplanes, 3-D occupancy; three-hidden-layer MLP, width 128 ($\approx 36{,}000$)
DeepWeightFlow (2026)	MNIST; MLP $[784, 32, 32, 10]$ (26,506)
Hyper-Representations (2022a)	MNIST; CNN-s with three convolutional and two fully connected layers (2,464)
RPG (2025)	CIFAR10; ResNet-18 (11,191,262)
p-diff (2024a)	CIFAR100; last two ResNet-18 BN γ, β tensors (2,048)
Tina (2024)	CIFAR100 in-distribution tasks; full ConvNet weights (5,370)
D2NWG (2025)	A 10-class task derived from ImageNet-1K; ResNet-18 10-way classification head ($D = 5{,}120$)
G.pt (2022)	MNIST; MLP (7,960)
Text2Weight (2025)	CIFAR100; rank-16 adapter and classification head (16,384)

F.2 Protocol II: Same-Setting Comparison

Protocol II uses the common 3,256-checkpoint KMNIST pool, TinyCNN$(16, 8)$ target with $D =$ 6,738, and 2,000-image WeightAUDIT set defined above. Each method uses five generator seeds and produces 256 generated checkpoints per seed. G.pt and Text2Weight are excluded for the incompatibilities stated in the main paper. Table 11 retains only method-specific adaptations and training settings. WINS receives labeled minibatches through DBS, with β fixed to the value reported for the configuration.

Together, Protocol I's original-setting audit and Protocol II's same-setting comparison provide complementary views. Under Protocol II, all compared methods can access the same training images and labels used to produce the checkpoint pool, and WINS introduces no additional or test data, providing a reasonably fair basis for comparison.

Table 13: Representative sampling-step configurations on KMNIST. Values are means over five generator seeds, and time is measured for generating 256 checkpoints on one RTX 5090. Shading follows the convention in the main paper.

K	Time for 256	Gen.	WDG	BDG	IX	UG	PD
1	0.01 s	0.820	0.68	1.50	1.51	1.91	0.16
4	0.02 s	0.812	0.77	1.64	1.48	1.32	0.50
16	0.08 s	0.797	0.87	1.70	1.44	1.26	0.75
400	1.96 s	0.769	0.96	1.72	1.62	1.16	0.99

G Sampling Efficiency and Resources

G.1 Number of Reverse Steps

The canonical sampler uses all $T = 400$ reverse steps. To study the quality-cost tradeoff, we also use a strided ancestral sampler with K effective reverse steps. Table 13 reports four representative settings. The one-step prediction is accurate but has low WDG and PD. At $K = 16$, WDG reaches 0.87 and PD reaches 0.75 in 0.08 seconds, while the full chain recovers the main-paper WINS profile.

G.2 Computational Resources

Constructing the experimental corpus of 3,256 TinyCNN checkpoints required 7.64 GPU-hours. We used 3,000 checkpoints to train WINS and reserved 256 for evaluation. Training WINS then required 209–225 s for 8,000 steps, or approximately 0.06 GPU-hours. These are offline preparation costs incurred before deploying the trained generator. Subsequent generation for the same task and architecture reuses this generator without rebuilding the checkpoint corpus or retraining WINS. Sampling also requires no optimization of the generated networks. We report these preparation costs separately from the 1.96 s required to generate a batch of 256 checkpoints.

All timings were measured on one RTX 5090. Training one TinyCNN network for 80 epochs and retaining its checkpoint takes 8.3–8.6 seconds. Peak training and sampling memory are 4.40 GiB and 0.44 GiB, respectively. Sampling times are reported in Table 13. The 1.96 s measurement uses all 400 reverse steps and corresponds to an amortized sampling time of approximately 0.0077 seconds per checkpoint. These measurements apply to the compact KMNIST setting.