Weight Space Representation Learning via Neural Field Adaptation

Anonymous Authors[1]

Abstract

In this work, we investigate the potential of weights to serve as effective representations, focusing on neural fields. Our key insight is that constraining the optimization space through a pre-trained base model and multiplicative low-rank adaptation (mLoRA) can induce structure in weight space. Across reconstruction, generation, and analysis tasks on 2D and 3D data, we find that mLoRA weights achieve high representation quality while exhibiting distinctiveness and semantic structure. When used with latent diffusion models, mLoRA weights enable higher-quality generation than existing weight-space methods.

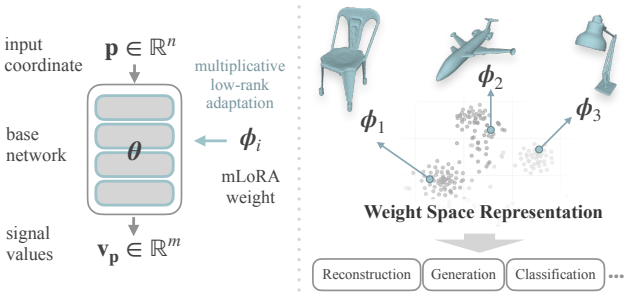

Figure 1. LoRA based weight space representation with neural fields. Given an input coordinate $\mathbf{p} \in \mathbb{R}^n$, a base neural field is adapted via mLoRA weights $\boldsymbol{\phi}_i$ to produce signal values $\mathbf{v_p} \in \mathbb{R}^m$, each weight representing an instance. The mLoRA weights themselves form structured representations in weight space, enabling diverse applications.

1. Introduction

Can neural network weights themselves serve as meaningful representations for data? We investigate this in the context of implicit neural representations (INRs), where networks are trained to overfit individual samples by mapping coordinates to values. INRs encode diverse modalities within a unified architecture (Xie et al., 2022), and since they inherently encode signals as parameters, using these weights as representations is a natural next step. However, network weights are inherently ambiguous: neuron permutations and scaling leave the network function unchanged (Zhao et al., 2025), so functionally identical networks can lie arbitrarily far apart in weight space, making the distribution multi-modal and hard to learn.

Our key insight is that introducing appropriate inductive biases on per-sample weights can transform these chaotic parameters into organized, semantic representations. To this end, as illustrated in Figure 1, we propose Low-Rank Adaptation (LoRA) (Hu et al., 2022) of a pre-trained base neural field. Two properties motivate this choice: first, LoRA constrains weight updates to a low-dimensional subspace defined by the base model, and subspace similarity analysis (Hu et al., 2022) shows adaptations at different ranks share common singular directions, evidencing a meaningful adaptation subspace; second, the low-rank constraint reduces representation dimensionality, mitigating the curse of dimensionality in parameter space.

We validate our approach across modalities and tasks: LoRA-based representations achieve lower reconstruction error, greater consistency across initializations, and better linear mode connectivity than standalone MLP weights; diffusion models trained on mLoRA weights outperform prior weight-space neural field generation; and on classification and clustering, the weight-space structure correlates with semantic properties of the encoded data.

2. Method

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$, we optimize one neural field per instance and represent each instance by the weights of its network. Since LoRA weights converge to a shared subspace (Hu et al., 2022), indicating structure imposed by the base network, we fine-tune a pre-trained base model via LoRA. For each $\mathbf{x}_i$, we optimize LoRA parameters $\boldsymbol{\phi}_i = \{\mathbf{A}_i^l, \mathbf{B}_i^l\}_{l=1}^L$ over L layers with base weights frozen:

$$\boldsymbol{\phi}_i = \min_{\boldsymbol{\phi}} \mathcal{L}_{\text{recon}} \left[f\left(\mathbf{p} \mid \text{LoRA}(\mathbf{W}, \boldsymbol{\phi})\right), \mathbf{x}_i(\mathbf{p}) \right] , \quad (1)$$

where $\mathbf{W}$ are the frozen base weights and $\mathcal{L}_{\text{recon}}$ is a reconstruction loss. The instance $\mathbf{x}_i$ is represented by $\boldsymbol{\phi}_i$ (Figure 1). All LoRAs share the same initialization.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review for the ICML 2026 Workshop on Weight-Space Symmetries. Do not distribute.

2.1. Multiplicative LoRA

We employ multiplicative LoRA rather than the standard additive formulation. Standard LoRA (Hu et al., 2022) adapts a pre-trained weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ through additive low-rank updates $\mathbf{W}' = \mathbf{W} + \mathbf{BA}$ where $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are low-rank matrices with rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$. We introduce a multiplicative formulation that applies weight updates through elementwise multiplication as $\mathbf{W}' = \mathbf{W} \odot \mathbf{BA}$, where $\odot$ denotes elementwise multiplication. This formulation enables more effective modulation of features, analogous to successful modulation techniques in generative neural fields (Anokhin et al., 2021; Karras et al., 2021; Chan et al., 2020). In our experiments, we demonstrate that this design is critical to obtaining good weight space structure and performance on reconstruction, generation, and discriminative tasks. We hypothesize this advantage stems from *feature entanglement* in the additive signal synthesis of neural fields (Yüce et al., 2022): additive LoRA injects new components into an already-entangled mixture, whereas multiplicative LoRA merely scales existing features, preserving channel structure.

2.2. Addressing Permutation Symmetry

Permutation symmetry refers to the invariance of network functions under neuron reordering, which causes functionally identical networks to occupy vastly different locations in weight space (Zhao et al., 2025; Gao & Others, 2024; Navon et al., 2023), making the distribution multi-modal and difficult to learn. Removing this symmetry collapses these modes into a canonical representation, yielding a smoother, more structured weight space.

Permutation symmetry has two sources in our setting. *External symmetries* from permutations of base network neurons are eliminated by sharing a single base model across all instances. *Internal symmetries* arise within the LoRA factors: both additive and multiplicative formulations permit permuting the r rank dimensions without changing the represented function (formal proofs in the supplementary material). Moreover, any $\mathbf{G} \in GL(r)$ gives $(\mathbf{AG})(\mathbf{G}^{-1}\mathbf{B}) = \mathbf{AB}$ (Lim et al., 2024a), so each function has a $GL(r)$-fold equivalence class in weight space.

To address internal symmetry, we apply the asymmetric masking technique of (Lim et al., 2024b) to all LoRA $\mathbf{A}$ matrices: in each row, $\sqrt{d_{\text{out}}}$ entries are randomly frozen (positions shared across instances). For standalone MLP and additive LoRA, frozen entries use higher-variance initialization $\mathcal{N}(0, \kappa\mathbf{I})$; for multiplicative LoRA, frozen entries are zeroed, gating off the corresponding rank component. The additive variant requires large κ to break symmetry effectively (Lim et al., 2024b), which is problematic for neural fields whose additive signal composition forces other weights to compensate for these fixed large magnitudes, entangling the representation. Multiplicative LoRA sidesteps this issue, which we validate empirically.

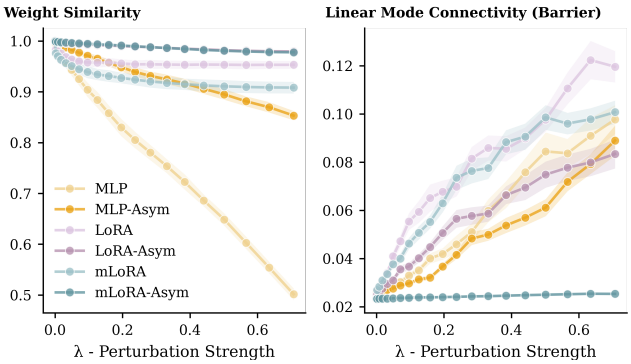

Figure 2. **Weight space structure analysis.** We measure weight similarity (cosine similarity) and the linear mode connectivity barrier (Chamfer distance) as a function of initialization perturbation strength λ. Each data point is averaged from 30 instances, the underlying shades are indicative of standard deviation.

Table 1. Weight space generation performance on 2D FFHQ.

	FD ↓	MMD-G ↓	MMD-P ↓
HyperDiffusion	0.241	0.158	1.887
MLP-Asym	0.287	0.203	2.423
LoRA	0.321	0.169	2.018
LoRA-Asym	0.269	0.157	1.877
mLoRA	0.100	0.056	0.674
mLoRA-Asym	**0.073**	**0.039**	**0.467**

3. Experiments

We conduct experiments to inspect the structure of the weight space and evaluate weight space representations across generation and discriminative tasks; reconstruction results are in the supplementary material.

Datasets. We evaluate on FFHQ (Karras et al., 2019) at 128×128 (higher than prior weight-space methods (Zhou et al., 2023b;a)) for 2D and on ShapeNet (Chang et al., 2015) for 3D, in both single-category (airplanes) and ten-category settings.

Candidate Representations. We compare six representations obtained by crossing three parameterizations (standalone *MLP*, additive *LoRA*, multiplicative *mLoRA*) with asymmetric masking on/off (*-Asym* suffix), isolating the effects of parameterization, operation type, and symmetry breaking. The baseline representation uses standalone *MLP* weights: for each instance $\mathbf{x}_i$, we fit a small MLP from scratch and take its parameters $\boldsymbol{\theta}_i$ as the representation, i.e. $\boldsymbol{\theta}_i = \min_{\boldsymbol{\theta}} \mathcal{L}_{\text{recon}}\left[f(\mathbf{p} \mid \boldsymbol{\theta}), \mathbf{x}_i(\mathbf{p})\right]$ with $\mathbf{p}$ a spatial coordinate. The architecture is a Fourier Feature layer followed by two linear layers. Following (Erkoç et al., 2023), all MLPs share the same random initialization across instances for consistency.

Table 2. Weight space generation performance on 3D ShapeNet. We examine a model trained on a single category *Airplane* and a 10-category model *Multi*.

	ShapeNet - Airplane						ShapeNet - Multi					
	mMD ↓	COV ↑	1-NNA ↓	FD ↓	MMD-G ↓	MMD-P ↓	mMD ↓	COV ↑	1-NNA ↓	FD ↓	MMD-G ↓	MMD-P ↓
HyperDiffusion	2.39	43.6%	78.2%	0.027	0.009	0.122	8.64	41.6%	78.3%	0.117	0.023	0.219
MLP-Asym	2.80	44.8%	80.9%	0.041	0.018	0.254	7.77	46.5%	74.0%	0.085	0.016	0.157
LoRA	116.4	3.1%	99.9%	1.553	0.669	7.163	152.9	10.9%	99.1%	1.014	0.319	2.501
LoRA-Asym	270.6	2.0%	100%	1.532	0.823	7.931	210.0	1.2%	100%	1.241	0.437	2.987
mLoRA	1.96	**46.2%**	**70.5%**	0.049	0.025	0.359	5.75	46.4%	61.9%	0.071	0.011	0.123
mLoRA-Asym	**1.89**	43.4%	71.9%	**0.011**	**0.003**	**0.041**	**5.52**	**49.6%**	**58.6%**	**0.026**	**0.004**	**0.040**

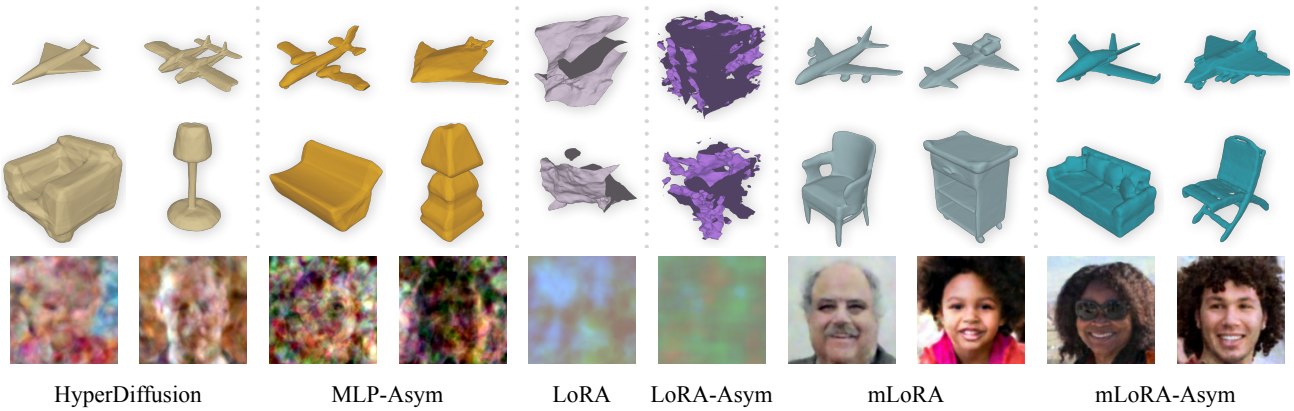

Figure 3. **Qualitative generation results.** Generated samples from diffusion models trained on different weight space representations. The top 2 rows show results generated by the *Airplane* model, followed by 2 rows from the *Multi*-class model. The bottom rows show 2D FFHQ generations.

3.1. Weight Space Structure Analysis

To study the geometric properties of each weight space, we conduct a stability analysis on the ShapeNet airplane model: for each instance we independently optimize two networks from different initializations $\boldsymbol{\iota}_1 \sim \mathcal{N}(0, \mathbf{I})$ and the variance-preserving perturbation $\sqrt{1-\lambda^2}\,\boldsymbol{\iota}_1 + \lambda\,\boldsymbol{\iota}_2$, where λ controls the perturbation strength. We report two metrics on the resulting weight pairs $(\boldsymbol{\phi}, \boldsymbol{\phi}_\lambda)$: (i) the cosine similarity, measuring whether different optimization paths converge to similar weight configurations, and (ii) the linear mode connectivity barrier (Frankle et al., 2020), measured as the Chamfer Distance between the ground-truth mesh and the mesh extracted (via marching cubes) from the averaged weights $(\boldsymbol{\phi} + \boldsymbol{\phi}_\lambda)/2$. Formal definitions are in the supplementary material. Figure 2 shows both metrics as a function of λ.

Results. Figure 2 shows LoRA and mLoRA improve weight similarity over standalone MLPs: MLP and MLP-Asym decay roughly linearly with perturbation strength, whereas LoRA-based representations saturate at large perturbations. LoRA does not, however, improve linear mode connectivity, consistent with the persistence of permutation symmetry in LoRA weights (Section 2.2).

The asymmetric mask improves both weight similarity and linear mode connectivity across parameterizations. mLoRA-Asym is exceptional: similarity stays very high and the barrier very low even under different initializations, indicating its weights **converge to a linear mode**. This is likely because multiplicative LoRA weights align with the base network once permutation symmetry is removed, as proven in Corollary 4 in the Supplementary.

Additive LoRA still lags behind mLoRA in linear mode connectivity even with asymmetric masking, consistent with the feature-entanglement argument of Section 2.1: multiplicative updates scale existing channels whereas additive updates mix new signals into an already-entangled representation.

3.2. Generation via Diffusion Models

We evaluate the generative capabilities of different weight space representations by training diffusion models on each parameterization. This experiment tests whether the learned weight spaces support high-quality generative modeling.

Implementation. For standalone MLP weights we reuse the HyperDiffusion (Erkoç et al., 2023) diffusion architecture; for LoRA weights we use our hierarchical LoRA-layer encoder (Section C.5). For standalone MLP weights (with and without asymmetric mask) we adopt the initialization of (Erkoç et al., 2023), fitting one instance and reusing its

weights to initialize the rest, which enables direct comparison with the state-of-the-art weight-space diffusion method.

Evaluation Metrics. For both 2D and 3D data we report the Fréchet Distance (FD) (Fréchet, 1957) and Maximum Mean Discrepancy with Gaussian (MMD-G) and polynomial (MMD-P) kernels, all computed on deep features (CLIP (Radford et al., 2021) for 2D, PointNet++ (Qi et al., 2017) for 3D). For 3D shapes we additionally report the Chamfer-distance based mMD, Coverage (COV), and 1-Nearest-Neighbor Accuracy (1-NNA) following (Erkoç et al., 2023). Formal definitions are in the supplementary material.

Results. Tables 1 and 2 report quantitative results on FFHQ, ShapeNet airplane, and ShapeNet ten-category; Figure 3 shows samples. mLoRA-Asym leads on nearly all metrics in 2D and 3D, producing diverse samples with high-frequency detail; this tracks its favorable weight-space structure, indicating weight-space geometry is crucial for diffusion-based generation. HyperDiffusion is competent on single-category airplanes but degrades sharply on the multi-category setting, suggesting difficulty modeling diverse weight distributions across classes. On FFHQ it fails to yield recognizable faces, whereas mLoRA and mLoRA-Asym succeed, marking the first successful weight-space generation for high-resolution natural images; prior work was limited to MNIST and CIFAR. LoRA and LoRA-Asym fail throughout, which we attribute to the additive-weight entanglement identified in our structure analysis.

3.3. Discriminative Tasks

To evaluate the distinctiveness and semantic structure of the learned weight representations, we conduct classification and clustering experiments on the ShapeNet ten-category dataset.

Classification. We evaluate two classification approaches. First, we use first nearest neighbor classification, which assigns each test weight to the category of its nearest neighbor in the training set using cosine similarity. Second, we train a linear classifier using logistic regression. All classifiers take the flattened weight representations as input and predict object categories.

Clustering. For clustering, we apply k-means with $k = 10$ (matching the number of categories) on the weight representations. We evaluate the clustering quality using the Adjusted Rand Index (ARI), which measures the agreement between the predicted clusters and the ground-truth categories while correcting for chance.

Results. Table 3 reports classification accuracy (two methods) and clustering Adjusted Rand Score across the six representations; Figure 4 shows t-SNE visualizations over ten categories, each point an instance colored by category. Semantic structure progresses clearly: LoRA beats standalone MLPs, mLoRA beats LoRA, and mLoRA is best overall, reaching 90% linear-classifier accuracy. Since mLoRA-Asym has better linear mode connectivity yet underperforms here, discriminative power is not directly tied to linear mode connectivity or permutation symmetry. All representations capture some semantic structure in t-SNE (same-category instances cluster), but only multiplicative LoRA weights show clear class separation, confirming that mLoRA weights capture semantic structure.

Table 3. Classification and clustering results on the ShapeNet ten-category dataset. We report the accuracy for two classification methods and the Adjusted Rand Score for clustering. All results are statistics from 10 runs with random data split and initializations.

	Clustering ARI ↑	1-NN ↑	Logistic ↑
MLP	39.3% ± 3.1%	50.0% ± 3.0%	78.1% ± 1.3%
MLP-Asym	48.4% ± 3.7%	46.8% ± 4.2%	82.1% ± 1.2%
LoRA	56.3% ± 3.3%	75.2% ± 1.7%	86.1% ± 1.1%
LoRA-Asym	47.4% ± 4.4%	59.1% ± 10.4%	84.3% ± 0.8%
mLoRA	**67.1**% ± 3.7%	**85.1**% ± 1.8%	**90.0**% ± 0.7%
mLoRA-Asym	56.5% ± 2.7%	80.8% ± 1.6%	84.5% ± 1.4%

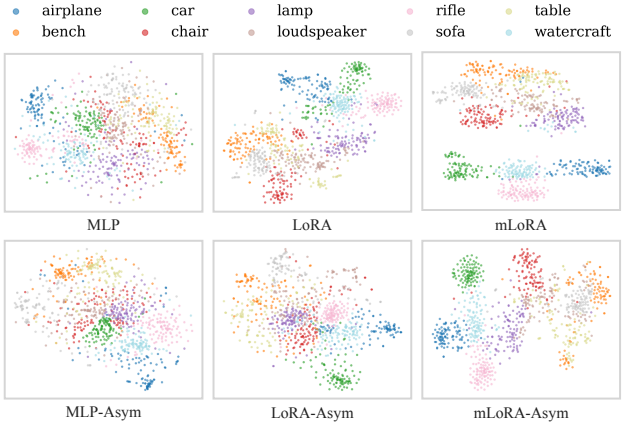

Figure 4. **t-SNE visualization of weight spaces.** Each point represents one instance from the ShapeNet ten-category dataset, colored by object category. Multiplicative LoRA weight spaces exhibit semantic structure.

4. Conclusion

We showed that independently optimized network weights can serve as effective data representations when shaped by appropriate inductive biases. Adapting a pre-trained base model via multiplicative LoRA turns chaotic parameters into structured, semantically organized representations that yield strong reconstruction, generation, and discriminative performance. Notably, mLoRA-Asym weights converge to a linear mode, and this weight-space geometry correlates strongly with diffusion-based generative performance.

References

Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., and Korzhenkov, D. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14278–14287, 2021.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Chan, E. R., Monteiro, M., Kellnhofer, P., Wu, J., and Wetzstein, G. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. arxiv e-prints, page. *arXiv preprint arXiv:2012.00926*, 2020.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, Y. and Wang, X. Transformers as meta-learners for implicit neural representations. In *European Conference on Computer Vision*, pp. 170–187. Springer, 2022.

Dravid, A., Gandelsman, Y., Wang, K.-C., Abdal, R., Wetzstein, G., Efros, A., and Aberman, K. Interpreting the weight space of customized diffusion models. *Advances in Neural Information Processing Systems*, 37:137334–137371, 2024.

Dupont, E., Goliński, A., Alizadeh, M., Teh, Y. W., and Doucet, A. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Dupont, E., Kim, H., Eslami, S. M. A., Rezende, D., and Rosenbaum, D. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, 2022.

Erkoç, Z., Ma, F., Öztireli, C., and Fua, P. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *International Conference on Computer Vision*, 2023.

Essakine, A., Cheng, Y., Cheng, C.-W., Zhang, L., Deng, Z., Zhu, L., Schönlieb, C.-B., and Aviles-Rivero, A. I. Where do we stand with implicit neural representations? a technical and performance survey. *arXiv preprint arXiv:2411.03688*, 2024.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Fréchet, M. Sur la distance de deux lois de probabilité. In *Annales de l'ISUP*, volume 6, pp. 183–198, 1957.

Gao, Y. and Others. Revisiting model merging: A statistical perspective. *arXiv preprint*, 2024.

Gordon, C., MacDonald, L. E., Saratchandran, H., and Lucey, S. D'oh: Decoder-only random hypernetworks for implicit neural representations. In *Proceedings of the Asian Conference on Computer Vision*, pp. 2507–2526, 2024.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations*, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jayasumana, S., Ramalingam, S., Veit, A., Glasner, D., Chakrabarti, A., and Kumar, S. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9307–9315, 2024.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021.

Klocek, S., Maziarka, Ł., Wołczyk, M., Tabor, J., Nowak, J., and Śmieja, M. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pp. 496–510. Springer, 2019.

Kofinas, M., Knyazev, B., Zhang, Y., Chen, Y., Burghouts, G. J., Gavves, E., Snoek, C. G. M., and Zhang, D. W. Graph neural networks for learning equivariant representations of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Lim, D., Gelberg, Y., Jegelka, S., Maron, H., et al. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024a.

Lim, D., Putterman, T., Walters, R., Maron, H., and Jegelka, S. The empirical impact of neural parameter symmetries, or lack thereof. *Advances in Neural Information Processing Systems*, 37:28322–28358, 2024b.

Luo, S. and Hu, W. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2837–2845, 2021.

Navon, A., Shamsian, A., Achituve, I., Fetaya, E., Chechik, G., and Maron, H. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pp. 25790–25816, 2023.

Ormaniec, O. and Others. Fusion of graph convolutional networks via optimal transport. *arXiv preprint*, 2025.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Peebles, W., Radosavovic, I., Brooks, T., Efros, A. A., and Malik, J. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2023.

Peebles, W. S. and Xie, S. Scalable diffusion models with transformers. 2023 ieee. In *CVF International Conference on Computer Vision (ICCV)*, volume 4172, 2022.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Singh, C. and Jaggi, M. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, 2020.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K., et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022.

Vyas, K., Humayun, A. I., Dashpute, A., Baraniuk, R. G., Veeraraghavan, A., and Balakrishnan, G. Learning transferable features for implicit neural representations. *Advances in Neural Information Processing Systems*, 37: 42268–42291, 2024.

Wang, K. and Others. Scaling weight space generative models. *arXiv preprint*, 2025.

Wang, K., Xu, Z., Zhou, Y., Zang, Z., Darrell, T., Liu, Z., and You, Y. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. Neural fields in visual computing and beyond. In *Computer graphics forum*, volume 41, pp. 641–676. Wiley Online Library, 2022.

Yüce, G., Ortiz-Jiménez, G., Besbinar, B., and Frossard, P. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19228–19238, 2022.

Zhao, B., Walters, R., and Yu, R. Symmetry in neural network parameter spaces. *arXiv preprint arXiv:2506.13018*, 2025.

Zhou, A., Yang, K., Burns, K., Cardace, A., Jiang, Y., Sokota, S., Kolter, J. Z., and Finn, C. Permutation equivariant neural functionals. *Advances in neural information processing systems*, 36:24966–24992, 2023a.

Zhou, A., Yang, K., Jiang, Y., Burns, K., Xu, W., Sokota, S., Kolter, J. Z., and Finn, C. Neural functional transformers. *Advances in neural information processing systems*, 36: 77485–77502, 2023b.

Zhou, L., Du, Y., and Wu, J. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5826–5835, 2021.

Supplementary Material

This supplementary material provides additional theoretical analysis, implementation details, and experimental results to support the main paper. Section A presents formal proofs demonstrating permutation symmetry in both additive and multiplicative LoRA parameterizations. Section B discusses related works in weight space learning and implicit neural representations. Section C provides comprehensive implementation details covering the dataset, the standalone MLP, the base model architecture and training, the diffusion model on weight representations, and the complete training pipeline. Section D formally defines the evaluation metrics used throughout our experiments. Section E reports reconstruction quality across the weight space representations. Section F presents an ablation study examining the effectiveness of the hierarchical LoRA layer encoder. Section G presents weight space interpolation experiments. Section H provides additional qualitative generation results on FFHQ and ShapeNet datasets. Finally, Section I discusses limitations and future research directions.

A. Permutation Symmetry in LoRA

We provide a formal proof that permutation symmetry exists within both additive and multiplicative LoRA parameterizations.

A.1. Permutation Symmetry in Additive LoRA

Theorem 1. *The adapted weight matrix from additive LoRA exhibits permutation symmetry with respect to the rank dimensions.*

Proof. Consider the additive LoRA formulation:

$$\mathbf{W}' = \mathbf{W} + \mathbf{BA} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$, and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$.

From a neural network perspective, the operation $\mathbf{BAx}$ for input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ can be viewed as a two layer network:

$$\mathbf{BAx} = \mathbf{B}(\mathbf{Ax}) \tag{3}$$

where $\mathbf{A}$ acts as an encoder layer compressing the input to r hidden activations, and $\mathbf{B}$ acts as a decoder layer expanding back to the output dimension.

Let $\mathbf{h} = \mathbf{Ax} \in \mathbb{R}^r$ denote the hidden activations. Consider a permutation matrix $\mathbf{P} \in \mathbb{R}^{r \times r}$ corresponding to permutation π. We can insert $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ between the two layers:

$$\mathbf{BAx} = \mathbf{BP}^T\mathbf{PAx} = (\mathbf{BP}^T)(\mathbf{PA})\mathbf{x} \tag{4}$$

Define $\tilde{\mathbf{A}} = \mathbf{PA}$ and $\tilde{\mathbf{B}} = \mathbf{BP}^T$. Then:

$$\mathbf{BA} = \tilde{\mathbf{B}}\tilde{\mathbf{A}} \tag{5}$$

This shows that $(\mathbf{B}, \mathbf{A})$ and $(\tilde{\mathbf{B}}, \tilde{\mathbf{A}})$ produce identical adapted weight matrices. Concretely, $\tilde{\mathbf{A}}$ permutes the rows of $\mathbf{A}$ (equivalently, permutes which hidden neuron each row corresponds to), and $\tilde{\mathbf{B}}$ permutes the columns of $\mathbf{B}$ by the same permutation (matching the hidden neuron reordering).

Since there are $r!$ possible permutations of r hidden neurons, and each permutation produces a functionally identical network, the additive LoRA weight space exhibits $r!$-fold permutation symmetry. □

Remark 1. This permutation symmetry is analogous to the well known permutation symmetry in standard MLPs (Zhou et al., 2023a): reordering hidden neurons (along with their corresponding incoming and outgoing weights) does not change the function computed by the network. In LoRA, the hidden dimension is the rank r, and permuting this dimension induces the symmetry. Figure 5(a) illustrates this symmetry, showing how the low rank matrices can be viewed as encoder and decoder layers where intermediate neurons can be reordered.

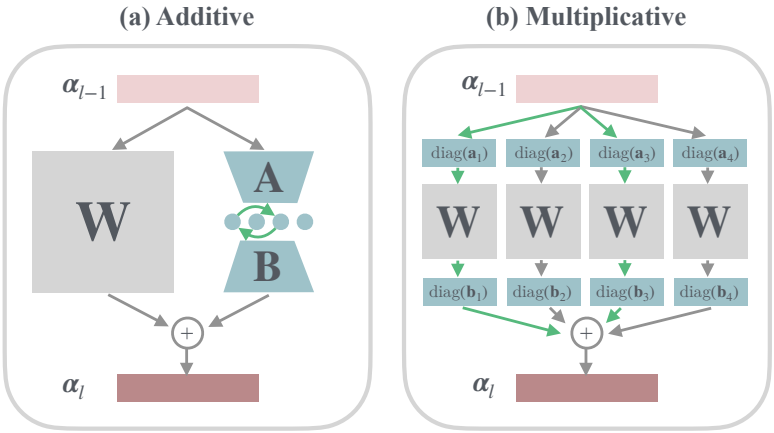

Figure 5. **Illustrating permutation symmetries within LoRA.** (a) Permutation symmetry in **additive** LoRA. Low rank matrices $\mathbf{A}$ and $\mathbf{B}$ could be seen as an encoder layer and a decoder layer. The order of the intermediate neurons could swapped without changing the output. (b) **multiplicative** LoRA could be seen as parallel pathways with different scaling factors for the input and output. The order of the pathways could be swapped without changing the output.

A.2. Permutation Symmetry in Multiplicative LoRA

Theorem 2. *The adapted weight matrix from multiplicative LoRA can be expressed as a sum of base weight matrices, each pre-multiplied and post-multiplied by diagonal matrices.*

Proof. Consider the multiplicative LoRA formulation from Section 2.1:

$$\mathbf{W}' = \mathbf{W} \odot \mathbf{BA} \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the base weight matrix, $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$, and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$.

We can decompose $\mathbf{B}$ and $\mathbf{A}$ into their column and row vectors respectively:

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_r \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_r^T \end{bmatrix} \tag{7}$$

where $\mathbf{b}_i \in \mathbb{R}^{d_{\text{out}}}$ and $\mathbf{a}_i \in \mathbb{R}^{d_{\text{in}}}$.

Therefore:

$$\mathbf{W}' = \mathbf{W} \odot \left(\sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T \right) = \sum_{i=1}^{r} \mathbf{W} \odot \left(\mathbf{b}_i \mathbf{a}_i^T \right) \tag{8}$$

For each term in the sum, the elementwise product $\mathbf{W} \odot (\mathbf{b}_i \mathbf{a}_i^T)$ can be expressed using diagonal matrices. Let $\text{diag}(\mathbf{b}_i)$ denote the diagonal matrix with $\mathbf{b}_i$ on the diagonal, and similarly for $\text{diag}(\mathbf{a}_i)$. Then:

$$\mathbf{W} \odot (\mathbf{b}_i \mathbf{a}_i^T) = \text{diag}(\mathbf{b}_i) \mathbf{W} \text{diag}(\mathbf{a}_i) \tag{9}$$

This can be verified by examining the (j, k) entry:

$$[\mathbf{W} \odot (\mathbf{b}_i \mathbf{a}_i^T)]_{jk} = W_{jk} \cdot (b_i)_j \cdot (a_i)_k \tag{10}$$

$$= (b_i)_j \cdot W_{jk} \cdot (a_i)_k \tag{11}$$

$$= [\text{diag}(\mathbf{b}_i) \mathbf{W} \text{diag}(\mathbf{a}_i)]_{jk} \tag{12}$$

Therefore:

$$\mathbf{W}' = \sum_{i=1}^{r} \text{diag}(\mathbf{b}_i)\mathbf{W}\text{diag}(\mathbf{a}_i) \tag{13}$$

This shows that the adapted weight matrix is a sum of terms, where each term is the base weight matrix pre-multiplied and post-multiplied by diagonal matrices constructed from the LoRA parameters. □

Corollary 3. *Permuting the rank indices $\{1, 2, \ldots, r\}$ with a permutation π does not change the adapted weight matrix $\mathbf{W}'$, as summation is commutative. This implies permutation symmetry in the LoRA weight space.*

This symmetry means that different configurations of LoRA parameters $\{\mathbf{a}_i, \mathbf{b}_i\}_{i=1}^{r}$ can represent the same function, making the weight space representation ambiguous without additional constraints. Figure 5(b) visualizes this symmetry by showing how multiplicative LoRA can be interpreted as parallel pathways that can be reordered without affecting the output.

Corollary 4. *Once permutation symmetry is eliminated, multiplicative LoRA weights are completely aligned with the channels in the base network.*

Proof. In Equation 13, each term $\text{diag}(\mathbf{b}_i)\mathbf{W}\text{diag}(\mathbf{a}_i)$ applies channel wise modulation to the base weight matrix $\mathbf{W}$. Specifically, $\text{diag}(\mathbf{a}_i)$ scales the input channels, while $\text{diag}(\mathbf{b}_i)$ scales the output channels. This operation preserves the channel structure of $\mathbf{W}$ through element wise scaling rather than mixing features across channels. When permutation symmetry is eliminated through techniques such as asymmetric masking, each rank component i is uniquely identified and cannot be arbitrarily reordered. In this regime, each pair $(\mathbf{a}_i, \mathbf{b}_i)$ corresponds to a specific modulation pattern applied to the base network channels. □

B. Related Work

B.1. Weight Space Learning

The treatment of neural network weights as learnable representations has emerged as a distinct research direction, progressing from early hypernetwork approaches to sophisticated weight-space generative models. Early work demonstrated that network parameters could be generated by auxiliary networks (Ha et al., 2016), though these methods suffered from prohibitive memory overhead when scaling to modern architectures (Wang & Others, 2025). The fundamental challenge of weight space manipulation stems from permutation symmetry: Functionally identical networks can have vastly different parameter configurations due to neuron reordering (Gao & Others, 2024; Navon et al., 2023).

Recent advances have addressed these challenges through multiple strategies. Model merging techniques leverage optimal transport and activation matching to align neurons before parameter fusion (Singh & Jaggi, 2020; Ormaniec & Others, 2025), while equivariant architectures explicitly respect weight-space symmetries when processing network parameters (Navon et al., 2023; Kofinas et al., 2024). A parallel research direction focuses on building neural networks that process weights as inputs. Neural Functional Transformers (Zhou et al., 2023b) and permutation-equivariant neural functionals (Zhou et al., 2023a) construct architectures that can extract information from network parameters while respecting their symmetries. Methods operating on Low-Rank Adaptation (LoRA) weights (Lim et al., 2024a) develop GL-equivariant networks to process low-rank weight spaces of fine-tuned models. Our work takes a different approach: rather than building model-agnostic weight encoders that process weights as external data (Zhou et al., 2023b;a; Navon et al., 2023; Lim et al., 2024a), we focus on enforcing structure directly on the weight space itself, through the choice of adaptation mechanism (multiplicative LoRA) and symmetry-breaking constraints (asymmetric masking). This makes the weights serve as effective representations without requiring additional encoding steps.

Generative modeling in weight space has seen significant progress, with diffusion models now capable of synthesizing functional neural networks. Weight-space generation with diffusion models (Wang et al., 2024; Wang & Others, 2025; Peebles et al., 2023) has been explored for generating models for image classification. Dravid *et al.* (Dravid et al., 2024) show that LoRA weights of diffusion models fine-tuned on human identities form an interpretable linear subspace, enabling semantic editing and sampling via PCA. With a different focus, our work explores neural field weights as representations of data that support high-quality generation and encode semantic structure.

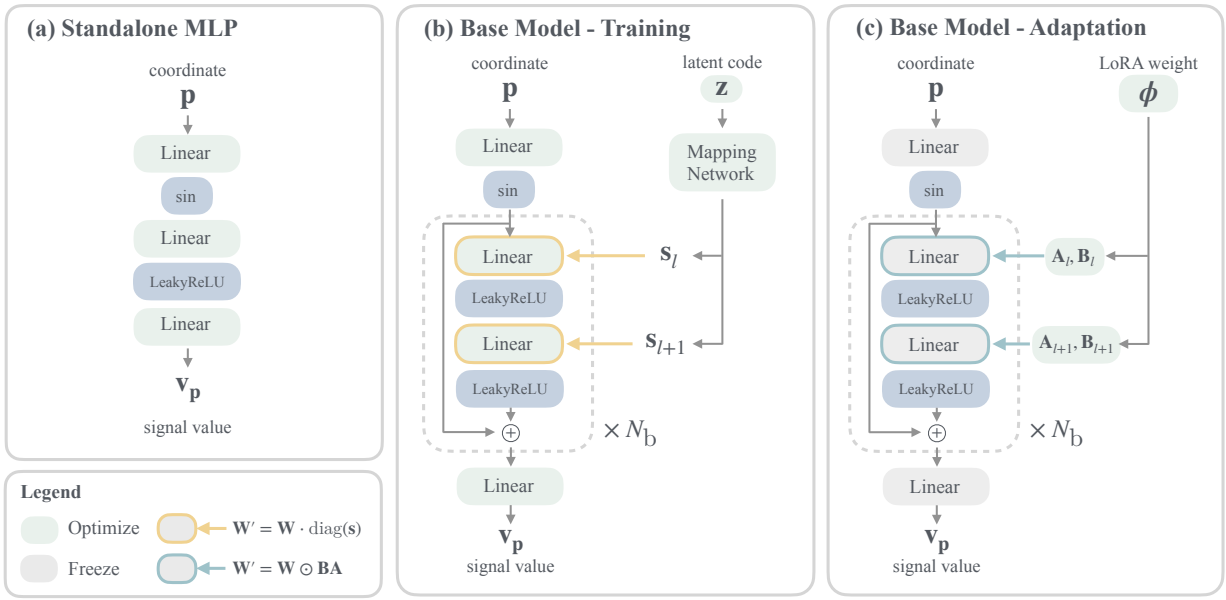

Figure 6. **Network architectures for weight space representations.** (a) Standalone MLP architecture with Fourier Feature layer followed by two linear layers. (b) Base model architecture with modulated fully connected layers. The network takes spatial coordinates $\mathbf{p}$ and style vector $\mathbf{s}$ as inputs, applying multiplicative weight modulation at each layer. (c) LoRA adaptation applied to the base model, where low rank matrices $\mathbf{A}$ and $\mathbf{B}$ adapt the frozen base weights.

B.2. Implicit Neural Representation

Implicit Neural Representations (INRs), also known as neural fields, are continuous functions parameterized by neural networks that map coordinates to signal values. INRs represent signals as continuous functions $\Phi : \mathbb{R}^n \to \mathbb{R}^m$, where a neural network maps n-dimensional coordinates to m-dimensional quantities. This paradigm enables resolution-independent and modality agnostic representations of complex signals (Xie et al., 2022; Essakine et al., 2024), employing identical network architectures across diverse modalities including 1D audio, 2D images, 3D shapes, and even 4D spatiotemporal data.

Beyond single-instance fitting, generalizable INRs have been proposed to learn priors across datasets through approaches based on autoencoders (Park et al., 2019), generative adversarial networks (GANs) (Anokhin et al., 2021; Karras et al., 2021; Chan et al., 2020) and shared layers (Vyas et al., 2024). The GAN-based works could be seen as extensions of the StyleGAN (Karras et al., 2019) paradigm into the realm of neural fields. They generate different instances by modulating an MLP trunk, which we use as the base network for fine-tuning.

Because INRs parameterize data as neural network functions, the weights offer a direct pathway to data representation. This perspective has practical applications in compression, with methods (Dupont et al., 2021; Gordon et al., 2024) demonstrating competitive compression ratios by storing quantized network parameters instead of raw data. However, whether the collection of weights could encode semantic structure remains an open question. Another line of work employs hypernetworks (Klocek et al., 2019) and transformers (Chen & Wang, 2022) to predict INR weights from input data via learned mappings as a way of data generation. Dupont *et al.* (Dupont et al., 2022) propose *functa*, which meta-learns a shared SIREN base network and represents each data point as a low-dimensional shift modulation vector for downstream tasks including generation and classification.

In contrast to all the above approaches, we investigate whether independently optimized weights can directly serve as meaningful representations. HyperDiffusion (Erkoç et al., 2023) trains a diffusion transformer to generate neural field weights as a means of synthesizing 3D shapes and 4D animated shapes. Our work builds on this to inspect factors that affect weight space generation performance and to explore semantic structures in neural field weights.

C. Implementation Details

C.1. Dataset

For FFHQ, we use the first $5,000$ samples from the dataset for all our experiments. For ShapeNet airplane, we use all $4,045$ samples. To create the ShapeNet 10-category dataset, we select the top 10 categories with the most samples and then randomly sample 500 instances from each category.

C.2. Standalone MLP

As shown in Figure 6(a), The standalone MLP is a Fourier Feature (Tancik et al., 2020) layer $\boldsymbol{\alpha}_1 = \sin(\omega_0 \cdot (\mathbf{W}_1\mathbf{p} + \mathbf{b}_1))$ followed by 2 linear layers.

- ω_0 is the frequency scaling factor for the Fourier Feature layer. For 2D FFHQ, we set $\omega_0 = 32$; For 3D ShapeNet, we set $\omega_0 = 1$. This value is chosen empiracally and shared with its LoRA-based counterparts.

- N_{hidden} is the number of hidden features in the linear layers. For the 2D FFHQ model, we use $N_{\text{hidden}} = 94$ for all layers; For the 3D ShapeNet, we use $N_{\text{hidden}} = 99$ for all layers, in order to have approximately the same number of learnable parameters as their LoRA-based counterparts.

As noted by Erkoç *et al.* (Erkoç et al., 2023), an initialization trick is employed to ensure diffusion model generalization. Specifically, an MLP with weights $\boldsymbol{\phi}_0$ is fitted to one instance from the dataset, and used as a shared initialization for all the rest of the fittings $\boldsymbol{\iota}_i = \boldsymbol{\phi}_0$. In other words, other instances are fitted by fine-tuning the weights from the first instance. This trick is crucial to the performance of HyperDiffusion, and therefore we employ it in the FFHQ and ShapeNet airplane experiments. However, in the multi-category experiment, it does not make sense to initialize a *chair* fitting with weights from an *airplane*, therefore we do not use this trick and instead use a shared random initilization for all the fittings, like is done for the LoRA-based weight representations.

For each instance, we run 10k steps, each step with $8,192$ points with an Adam optimizer, where learning rate adaptively decay from 10^{-2} to 10^{-5}. The same optimizer and hyperparameters are used for the LoRA-based weight representations.

C.3. Base Model Architecture

For LoRA-based representations, we require a strong base model that captures transferable features across the data distribution. We adopt a coordinate-based neural field architecture with multiplicative weight modulation, a design found across multiple generative neural field works (Anokhin et al., 2021; Karras et al., 2021; Chan et al., 2020), as illustrated in Figure 6(b). The modulation mechanism naturally aligns with our multiplicative LoRA formulation, and the architecture is applicable, but not limited, to 2D and 3D data.

The architecture consists of a mapping network and a synthesis network. The mapping network is a multilayer perceptron that transforms a latent code $\mathbf{z} \in \mathbb{R}^{d_z}$ into intermediate style vectors $\{\mathbf{s}_l\}_{l=1}^{L}$, where L is the number of synthesis layers. For 2D FFHQ, we use $d_z = 256$ and an 8 layer mapping network with hidden dimension 256. For 3D ShapeNet, we use $d_z = 128$ and a 4 layer mapping network with hidden dimension 256.

The synthesis network takes spatial coordinates $\mathbf{p}$ as input and produces signal values through a sequence of synthesis blocks, each block contains 2 modulated fully connected. The blocks are connected with residual connections to ensure gradient flow. Each layer l first applies weight modulation, where the weight matrix $\mathbf{W}_l$ is scaled by a learned affine transformation of the style vector: $\mathbf{W}'_l = \mathbf{W}_l \cdot \text{diag}(\mathbf{s}_l)$, where $\mathbf{s}_l = \mathbf{A}_l\mathbf{s} + \mathbf{b}_l$ is computed from the style vector via an affine transformation. The modulated weights are then normalized per output channel as $w'_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k}(w'_{ijk})^2 + \epsilon}$, where $\epsilon = 10^{-8}$ for numerical stability. The layer then computes activations as $\boldsymbol{\alpha}_{l+1} = \text{ReLU}(\mathbf{W}'_l\boldsymbol{\alpha}_l + \mathbf{b}_l)$. For 2D FFHQ, we use 6 synthesis blocks with channel dimensions 256. For 3D ShapeNet, we use 4 synthesis blocks with channel dimensions 512.

C.4. Base Model Training

We train the base model using the variational autodecoder paradigm (Park et al., 2019). This training scheme is desirable because it requires no encoder design, aligning with the data-agnostic quality of INRs. Given a dataset $\{\mathbf{x}_i\}_{i=1}^{N}$, we jointly

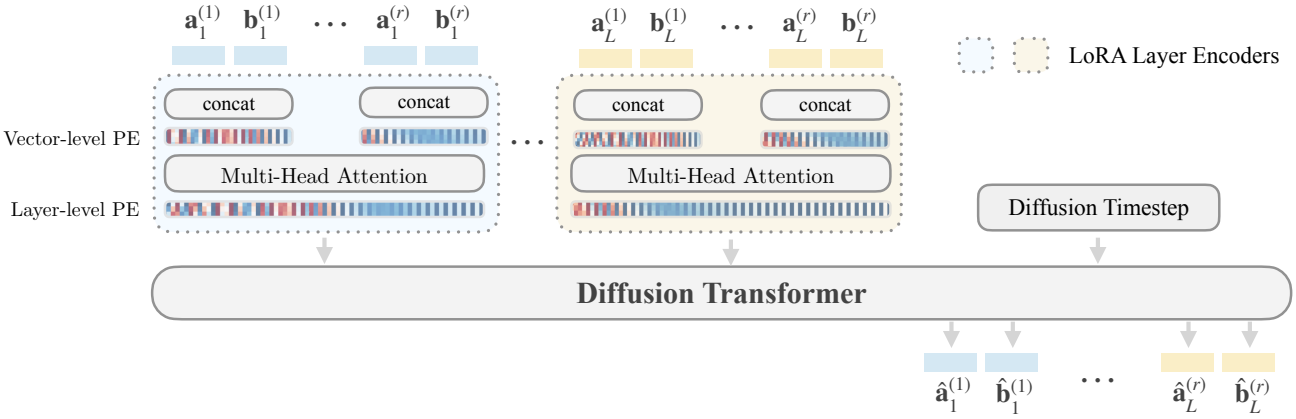

Figure 7. **Diffusion Transformer with hierarchical LoRA layer encoder architecture.** For each layer l, we treat vector pairs $(\mathbf{a}_l^{(i)}, \mathbf{b}_l^{(i)})$ as tokens. Vector-level positional encodings capture rank dimension indices, followed by multi-head attention that models interactions among the r rank components within the layer. This hierarchical design enables the model to learn both local (within-layer) dependencies among rank components and global (cross-layer) relationships across different layers of the neural field.

optimize the network parameters $\boldsymbol{\theta}$ and per-instance latent codes $\{\mathbf{z}_i\}_{i=1}^N$ by solving

$$\min_{\boldsymbol{\theta}, \{\mathbf{z}_i\}} \sum_{i=1}^{N} \mathcal{L}_{\text{recon}}(f_{\boldsymbol{\theta}}(\mathbf{p}, \mathbf{z}_i), \mathbf{x}_i(\mathbf{p})) + \lambda_r \|\mathbf{z}_i\|_2^2 \,, \tag{14}$$

where $\mathbf{p}$ represents spatial coordinates and λ_r controls the latent code regularization.

We devise a multistage progressive training strategy that gradually increases sampling resolution while decreasing batch size. Early stages use large batch sizes with low resolution (batch size of 256 with $2,048$ points per instance) to establish the latent code manifold, while late stages use small batch sizes with high resolution (batch size of 16 with $32,768$ points per instance) to capture fine details. This strategy ensures stable latent code initialization while maintaining computational efficiency.

We train 350k steps with an Adam optimizer, where the learning rate gradually decay from 10^{-3} to 10^{-5} in 5 stages. For regularization factor we use $\lambda_r = 10^{-4}$ Exponential moving average is applied on the base model weights.

C.5. Diffusion Model on Weight Representations

To evaluate the potential of the weight representations in generative tasks, we train diffusion models to learn their distribution. Following the DDPM framework, we define a forward diffusion process that gradually adds Gaussian noise to the weight representations, i.e.

$$q(\boldsymbol{\phi}_t|\boldsymbol{\phi}_{t-1}) = \mathcal{N}(\boldsymbol{\phi}_t; \sqrt{1-\beta_t}\boldsymbol{\phi}_{t-1}, \beta_t \mathbf{I}) \,, \tag{15}$$

where $\boldsymbol{\phi}$ represents the flattened weight parameters (either full MLP weights or LoRA matrices), and β_t follows a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 2 \times 10^{-2}$ over $T = 500$ timesteps.

We parameterize the reverse process using a diffusion transformer (DiT) (Peebles & Xie, 2022) that predicts the noise added to the weights. For standalone MLP weights, we adopt the architecture from (Erkoç et al., 2023; Peebles et al., 2023). For LoRA weights, we design a hierarchical LoRA layer encoder module that respects the structural properties of low-rank weight matrices, shown in Figure 7. Each layer l is processed as follows. First, we treat each vector pair $(\mathbf{a}_l^{(i)}, \mathbf{b}_l^{(i)})$ as a token, where $\mathbf{a}_l^{(i)}$ and $\mathbf{b}_l^{(i)}$ are the i-th columns and rows of matrices $\mathbf{A}^{(l)}$ and $\mathbf{B}^{(l)}$, respectively. Vector-level positional encodings are then applied to capture the rank dimension index. A multi-head attention module with r attention heads encodes the interactions among the r vector pairs within the layer, allowing the model to learn dependencies between different rank components. Finally, layer-level positional encodings are applied to the aggregated layer representation before feeding into the main transformer.

This hierarchical design is motivated by the compositional structure of LoRA weights. Within each layer, the low-rank decomposition creates dependencies among the r rank components, which the intra-layer attention module explicitly models.

Different layers, however, operate at different semantic levels in the neural field, making layer-level encoding essential for capturing cross-layer relationships. This architecture naturally respects the paired nature of LoRA matrices while enabling the model to learn both local (within-layer) and global (cross-layer) weight space structure.

With the noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{\phi}_t, t)$, we optimize the simplified diffusion objective

$$\mathcal{L} = \mathbb{E}_{t\sim\mathcal{U}(1,T),\boldsymbol{\phi}_0\sim p_{\text{data}},\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\nu}}(\boldsymbol{\phi}_t, t)\|^2\right] . \tag{16}$$

During inference, we use DDIM sampling with 100 steps to efficiently sample new weight representations, which are then used to instantiate novel neural fields. The diffusion transformer has 2880 hidden size (i.e., the size of each token after linear projection or layer encoder), 12 layers, and 16 self-attention heads. We train the diffusion transformer with batch size of 256 and learning rate of 2×10^{-4} for 6000 epochs until convergence.

C.6. Pipeline

Algorithm 1 describes the complete pipeline for weight space representation learning and generation. The process consists of three stages: First, we train a base model using variational autodecoding for LoRA based representations. Second, we construct a dataset of weight representations by fitting neural fields to individual instances. For standalone MLP, an initialization trick is employed where one instance is first fitted and then used to initialize all other fittings. For LoRA based representations, all instances share the same random initialization. Third, we train a diffusion model on the collected weight representations to enable generation of novel instances.

D. Evaluation Metrics for Generation

We calculate distributional metrics for both 2D and 3D. For 3D, we also calculate distance-based metrics. All metrics are calculated between $2,048$ generated samples and $2,048$ reference samples.

D.1. Distributional Difference

These metrics operate on features extracted by deep learned models, as deep learned feature extractors project the data into semantically rich embedding spaces where distances better correlate with human perception of similarity. We employ these metrics in their mathematical form rather than using modality-specific implementations like FID (Heusel et al., 2017) or KID (Bińkowski et al., 2018), as this allows for consistent evaluation across different data modalities. For 2D images, we use CLIP (Radford et al., 2021) as the feature extractor; for 3D shapes, we use a PointNet++ (Qi et al., 2017).

Given generated distribution P and reference distribution Q, to make the metric more comparable, we first normalize the extracted features by

$$\boldsymbol{\rho} \leftarrow \frac{\boldsymbol{\rho} - \mu_Q}{\sigma_Q}$$

for both the generated set and reference set, where μ_Q and σ_Q are the scalar mean and standard deviation calculated from the reference set.

Fréchet Distance (FD) (Fréchet, 1957) measures the distance between two multivariate Gaussian distributions fitted to feature representations of generated and reference samples. Given feature representations from a pretrained network, we compute the mean $\boldsymbol{\mu}_P$ and covariance $\boldsymbol{\Sigma}_P$ for generated distribution P and mean $\boldsymbol{\mu}_Q$ and covariance $\boldsymbol{\Sigma}_Q$ for reference distribution Q. The Fréchet Distance is then computed as:

$$\begin{aligned}\text{FD}(P, Q) = \frac{1}{N_{\text{feature}}}[\|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|_2^2 + \\ \text{Tr}\left(\boldsymbol{\Sigma}_P + \boldsymbol{\Sigma}_Q - 2(\boldsymbol{\Sigma}_P\boldsymbol{\Sigma}_Q)^{1/2}\right)] .\end{aligned}$$

where N_{feature} is the feature dimension of the feature extractor.

Maximum Mean Discrepancy (MMD) with respect to a positive definite kernel ψ is defined by:

$$\begin{aligned}\text{MMD}(P, Q) = \mathbb{E}_{\mathbf{x},\mathbf{x}'}[\psi(\mathbf{x}, \mathbf{x}')] + \mathbb{E}_{\mathbf{y},\mathbf{y}'}[\psi(\mathbf{y}, \mathbf{y}')] \\ - 2\mathbb{E}_{\mathbf{x},\mathbf{y}}[\psi(\mathbf{x}, \mathbf{y})],\end{aligned}$$

where $\mathbf{x}, \mathbf{x}' \sim P$ are samples from the generated distribution and $\mathbf{y}, \mathbf{y}' \sim Q$ are samples from the reference distribution. This metric does not make the multivariate Gaussian assumption and is reported to be more reliable and sample efficient (Jayasumana et al., 2024; Bińkowski et al., 2018). We calculate this metric with 2 types of kernels.

1. Polynomial kernel (MMD-P)
   $\psi_p(\mathbf{x}, \mathbf{y}) = (\gamma_p \cdot \mathbf{x}^T\mathbf{y} + c)^d$ with degree $d = 3$ and offset $c = 1$. Following the practice of KID (Bińkowski et al., 2018), we choose $\gamma_p = 1/N_{\text{feature}}$.
2. Gaussian RBF kernel (MMD-G)
   $\psi_g(\mathbf{x}, \mathbf{y}) = \exp(-\gamma_g \|\mathbf{x} - \mathbf{y}\|_2^2)$ with $\gamma_g = 1/(2\sigma_g^2)$; we choose $\sigma_g = N_{\text{feature}}$.

D.2. Distance-based Metrics for 3D Shapes

For 3D shapes, we calculate distance-based metrics following (Erkoç et al., 2023; Vahdat et al., 2022; Luo & Hu, 2021; Zhou et al., 2021). We denote the distance function as $D(\mathbf{x}, \mathbf{y})$ for the Chamfer Distance between two shapes $\mathbf{x}$ and $\mathbf{y}$. The metrics are defined as:

$$\text{mMD}(P, Q) = \mathbb{E}_{\mathbf{y} \sim Q} \left[\min_{\mathbf{x} \sim P} D(\mathbf{x}, \mathbf{y}) \right],$$
$$\text{COV}(P, Q) = \frac{|\{\arg\min_{\mathbf{y} \sim Q} D(\mathbf{x}, \mathbf{y}) | \mathbf{x} \sim P\}|}{|Q|},$$
$$\text{1-NNA}(P, Q) = \frac{\sum_{\mathbf{x} \sim P} \mathbb{1}[\mathbf{N}_\mathbf{x} \sim P] + \sum_{\mathbf{y} \sim Q} \mathbb{1}[\mathbf{N}_\mathbf{y} \sim Q]}{|P| + |Q|},$$

where in the 1-NNA metric $\mathbf{N}_\mathbf{x}$ is the shape that is closest to $\mathbf{x}$ in both generated and reference distributions, that is,

$$\mathbf{N}_\mathbf{x} = \arg\min_{\mathbf{z} \sim P \cup Q} D(\mathbf{x}, \mathbf{z}).$$

For mMD, lower is better; for COV, higher is better; for 1-NNA, 50% is optimal.

E. Reconstruction Experiments

The reconstruction task directly corresponds to the fitting procedure described in Section 2, where each weight space representation is optimized to reconstruct individual instances. For 2D images, we measure the PSNR (higher is better). For 3D shapes, we measure the Chamfer Distance (lower is better). We also report the number of learnable parameters for each representation.

Table 4. Reconstruction quality. We report the PSNR for 2D FFHQ, the Chamfer Distance $\times 10^{-2}$ for 3D ShapeNet, and the number of trainable parameters in each weight space representation. Parameters frozen by the asymmetric mask are reduced from the count.

	FFHQ		**ShapeNet**		
	PSNR ↑	# Params	CD-A ↓	CD-M ↓	# Params
MLP	35.11	27,357	2.57	3.78	30,196
MLP-Asym	33.28	24,537	2.64	4.00	27,226
LoRA	35.69	27,395	2.44	3.39	29,696
LoRA-Asym	24.63	26,307	2.46	3.44	27,539
mLoRA	35.65	27,395	2.45	3.49	29,696
mLoRA-Asym	**36.91**	26,307	**2.41**	**3.35**	27,539

Results. Table 4 shows that LoRA, mLoRA, and mLoRA-Asym reach better reconstruction with compact parameter counts than MLP-based representations, which we attribute to transferable features in the base network that fine-tuning leverages with few adaptation parameters. Multiplicative LoRA outperforms its additive counterpart, echoing multiplicative modulation in generative neural fields (Anokhin et al., 2021; Chan et al., 2020; Karras et al., 2021). Surprisingly, mLoRA-Asym beats mLoRA despite frozen parameters; we hypothesize the mask reduces parameter entanglement. LoRA-Asym performs poorly on FFHQ, likely due to entanglement from the large frozen-weight variance $\kappa = 6$ (empirically determined), as discussed in Section 2.2.

Table 5. Ablation study of LoRA Layer Encoder on 3D ShapeNet - 10 category.

	ShapeNet - Multi					
	mMD ↓	COV ↑	1-NNA ↓	FD ↓	MMD-G ↓	MMD-P ↓
w/o	**5.18**	47.7%	61.2%	0.049	0.008	0.098
with	5.52	**49.6%**	**58.6%**	**0.026**	**0.004**	**0.040**

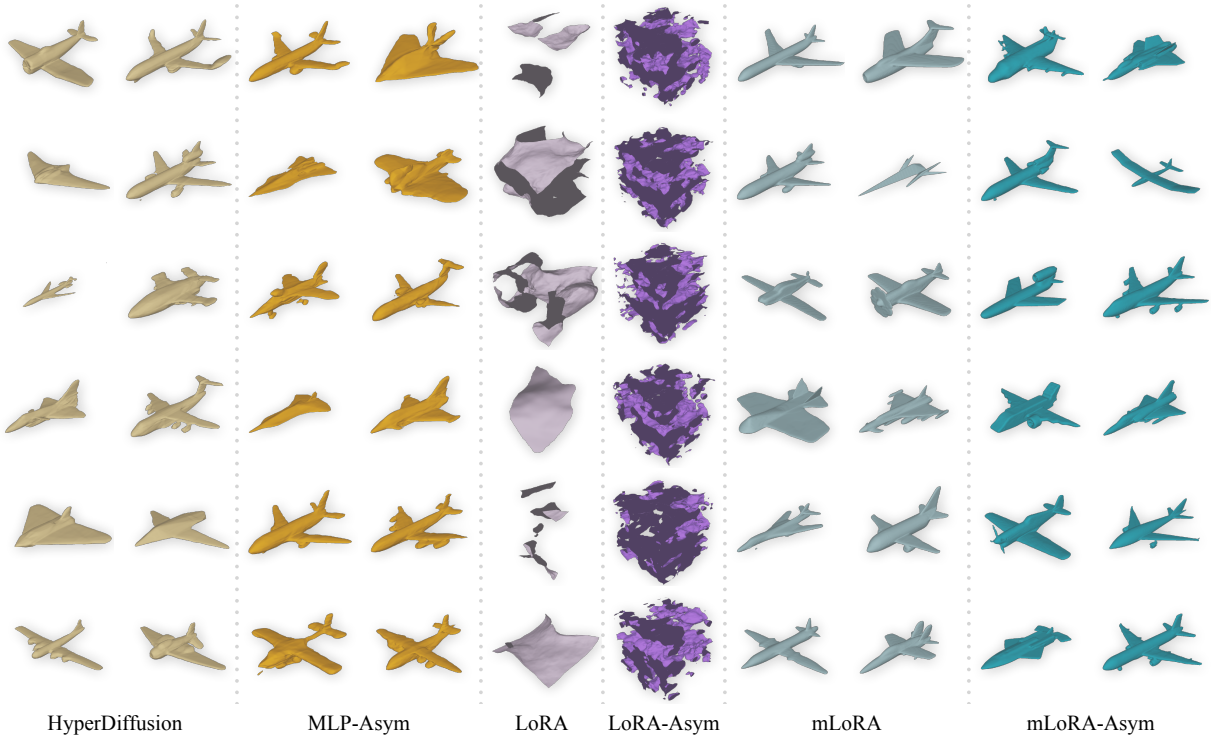

Figure 8. **Additional qualitative generation results on ShapeNet - Airplanes.**

F. Ablation Study

We examine the effectiveness of the hierarchical LoRA layer encoder introduced in Section C.5. To isolate its contribution, we train a baseline diffusion model without the layer encoder on the ShapeNet multi-category dataset. This baseline treats each weight matrix as an independent token, directly feeding flattened LoRA matrices into the transformer without the hierarchical processing that models within layer rank dependencies and cross layer relationships.

Table 5 compares the baseline against our full model with the hierarchical layer encoder. The results demonstrate that the layer encoder provides substantial improvements across all metrics. The full model achieves higher coverage (49.6% vs 47.7%), better 1-NNA (58.6% vs 61.2%), and significantly better distributional metrics: FD improves from 0.049 to 0.026, MMD-G from 0.008 to 0.004, and MMD-P from 0.098 to 0.040. These improvements confirm that explicitly modeling the compositional structure of LoRA weights, where rank components within each layer interact and different layers encode different semantic levels, is essential for effective weight space generation. The hierarchical design enables the diffusion model to respect the intrinsic organization of neural field weights, leading to higher quality generated samples.

G. Weight Space Interpolation

Figure 12 visualizes linear interpolations between pairs of mLoRA weight representations on FFHQ. We linearly interpolate between two instances' LoRA weight pairs $(\mathbf{A}_1, \mathbf{B}_1)$ and $(\mathbf{A}_2, \mathbf{B}_2)$, evaluating the resulting neural field at each interpolation step. While the interpolated weights do not always produce smooth, perceptually gradual transitions between instances (as would be expected from a continuous learned latent space), this does not undermine our claims about the quality of the

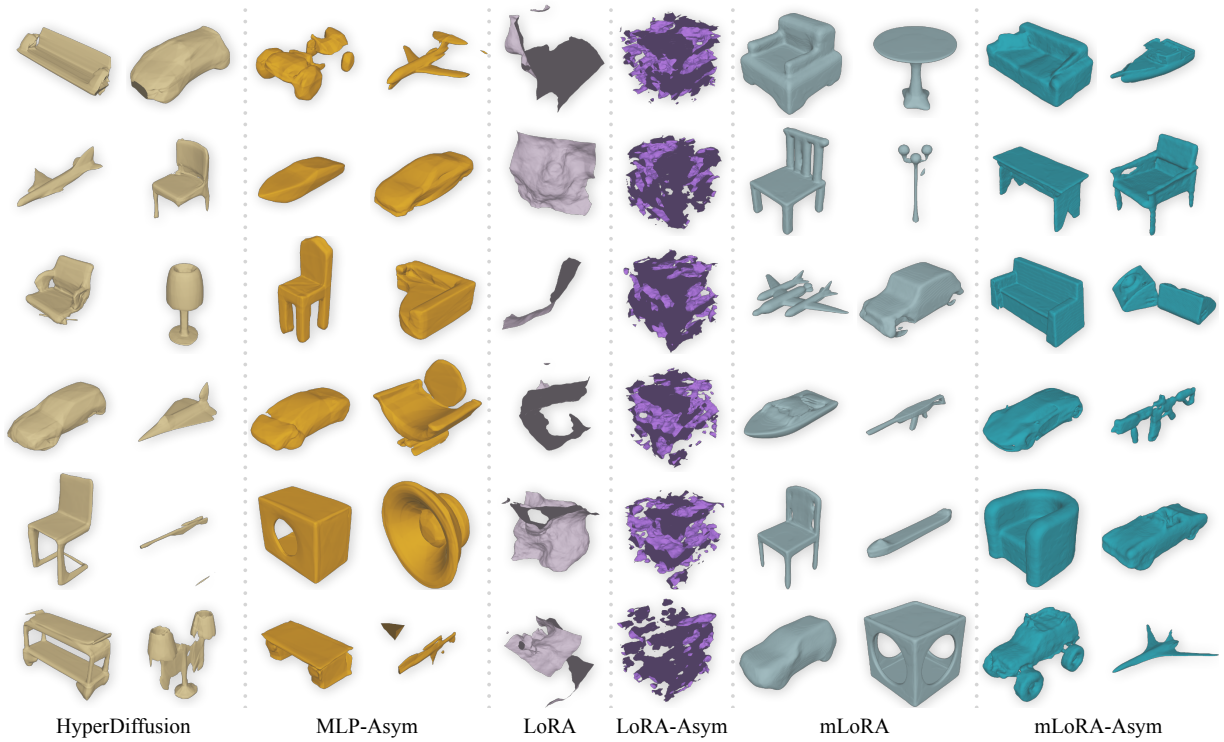

Figure 9. **Additional qualitative generation results on ShapeNet - Multi.**

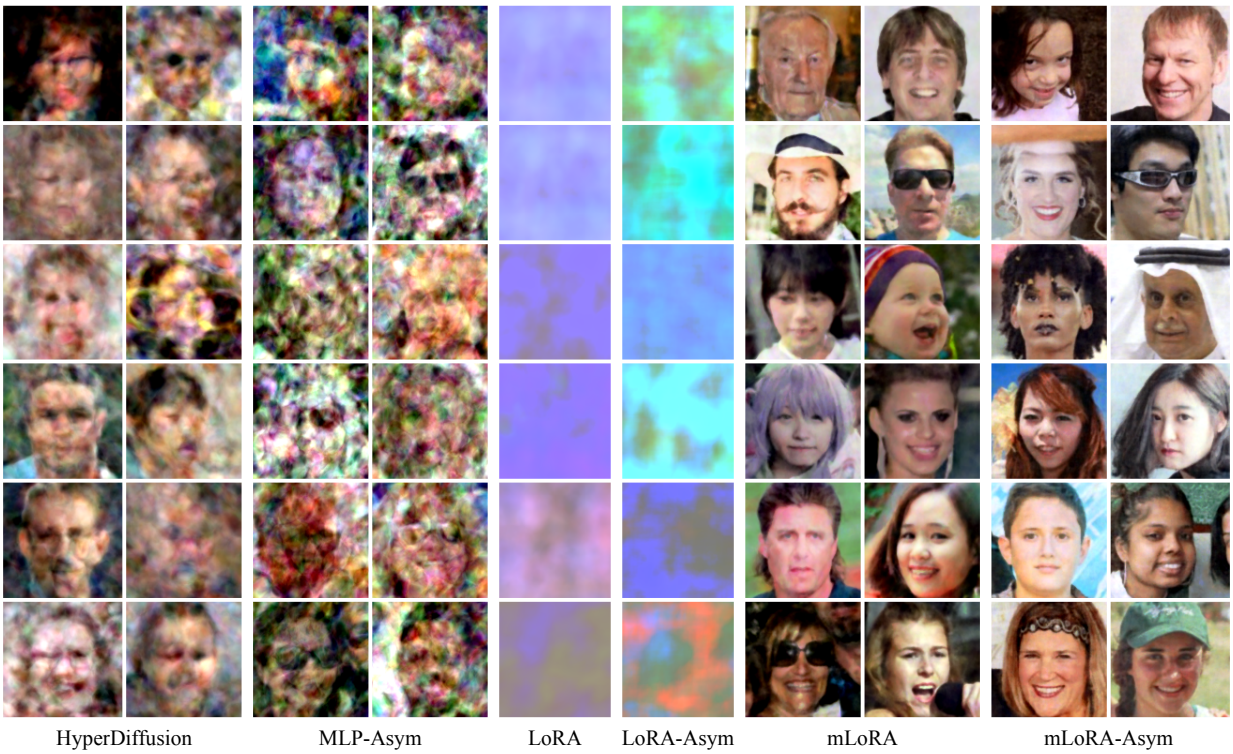

Figure 10. **Additional qualitative generation results on FFHQ.**

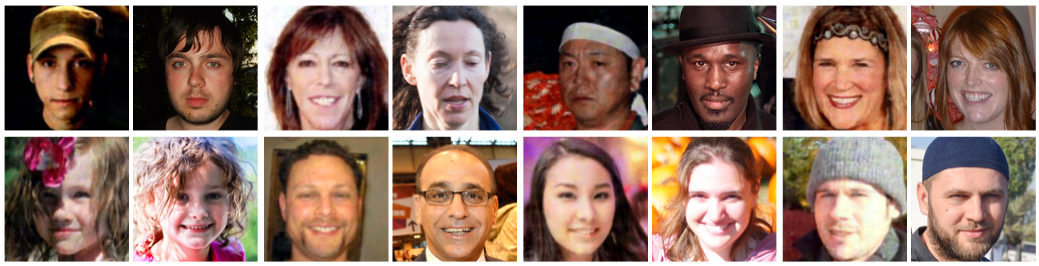

Figure 11. **Novelty check.** For each generated FFHQ image (left), we show its nearest neighbor from the training set retrieved by CLIP feature similarity (right). Generated samples are visually distinct from training data, confirming generalization rather than memorization.

weight space representations. Smooth interpolation is characteristic of continuous latent spaces specifically optimized for this property, such as VAE latent codes, but structured representations like VQ-VAE quantized codes and point-cloud latents also lack this property yet achieve strong generative performance (Vahdat et al., 2022). Our experiments demonstrate that mLoRA weights support high-quality generation (Tables 2–3 in the main paper) and exhibit clear semantic structure for classification (Table 4, Figure 5 in the main paper), which are the primary criteria for effective weight space representations.

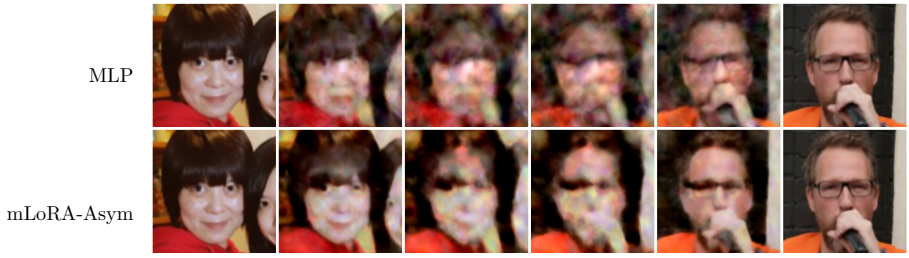

Figure 12. **Weight space interpolation.** Linear interpolation between pairs of mLoRA-Asym weight representations on FFHQ. Columns show the two endpoint instances (leftmost, rightmost) and intermediate interpolated reconstructions.

H. Additional Qualitative Results

We provide additional qualitative results on diffusion generation. Please see Figure 8 for results on ShapeNet Airplanes, Figure 9 for results on ShapeNet Multi, and Figure 10 for results on FFHQ.

To verify that generated samples are novel rather than memorized reproductions of training data, we perform a CLIP-based nearest-neighbor analysis on FFHQ generations from the mLoRA-Asym configuration. For each generated image, we retrieve its nearest neighbor from the training set using CLIP feature similarity. Figure 11 shows paired comparisons (left: generated, right: nearest training neighbor). The generated images are visually distinct from their nearest training neighbors, confirming that the diffusion model generalizes rather than memorizes.

I. Limitation and Future Work

While our work establishes that neural network weights can serve as effective data representations, several limitations present opportunities for future research.

First, our approach requires all instances to share the same pre-trained base model and initialization. This is a practical limitation: the base model may not suit all INR architectures, and meaningful weight space comparisons between instances trained on different base models are not possible. Future work could explore methods to reduce this requirement or to align weight spaces across different base models.

Second, our approach requires finetuning a base-model which is computationally more expensive than fitting small MLPs. This requirement prevents evaluation on datasets with hundreds of thousands of samples, constraining our experiments to datasets with thousands of instances. Future work could explore methods to eliminate this requirement, or develop more computationally efficient multiplicative LoRA adaptation procedures.

Third, while our method achieves the first successful weight space generation on relatively high resolution natural images and demonstrates superior performance compared to prior weight space methods, the generation quality does not yet match state-of-the-art image generative models such as latent diffusion models. Future work could focus on closing this performance gap while preserving data modality agnosticism.

Algorithm 1 Pipeline for weight space learning and generation.

Input: Dataset $\{\mathbf{x}_i\}_{i=1}^N$, parameterization type $\tau \in \{\text{MLP}, \text{LoRA}, \text{mLoRA}\}$
Output: Trained diffusion model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$

Stage 1: Base Model Training (for LoRA/mLoRA only)
if $\tau \in \{\text{LoRA}, \text{mLoRA}\}$ **then**
  Initialize base model weights $\boldsymbol{\theta}$ and latent codes $\{\mathbf{z}_i\}_{i=1}^N$
  Jointly optimize $\boldsymbol{\theta}$ and $\{\mathbf{z}_i\}$ via variational autodecoding:
  $\boldsymbol{\theta}^*, \{\mathbf{z}_i^*\} \leftarrow \arg\min \sum_{i=1}^N \mathcal{L}_{\text{recon}}(f(\mathbf{p}, \mathbf{z}_i|\boldsymbol{\theta}), \mathbf{x}_i(\mathbf{p})) + \lambda_r \|\mathbf{z}_i\|_2^2$
  Freeze base model: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$
end if

Stage 2: Instance Fitting
if $\tau = \text{MLP}$ **then**
  Fit one instance:
  $\boldsymbol{\phi}_0 \leftarrow \arg\min_{\boldsymbol{\phi}} \mathcal{L}_{\text{recon}}(f(\mathbf{p} \mid \boldsymbol{\phi}), \mathbf{x}_1(\mathbf{p}))$
  for $i = 2$ to N **do**
    Initialize: $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\phi}_0$
    Optimize:
    $\boldsymbol{\phi}_i \leftarrow \arg\min_{\boldsymbol{\phi}} \mathcal{L}_{\text{recon}}(f(\mathbf{p} \mid \boldsymbol{\phi}), \mathbf{x}_i(\mathbf{p}))$
  end for
else
  Sample shared random initialization: $\boldsymbol{\iota}_0 \sim \mathcal{N}(0, \mathbf{I})$
  for $i = 1$ to N **do**
    Initialize: $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\iota}_0$
    Optimize:
    $\boldsymbol{\phi}_i \leftarrow \arg\min_{\boldsymbol{\phi}} \mathcal{L}_{\text{recon}}(f(\mathbf{p} \mid \text{LoRA}(\mathbf{W}, \boldsymbol{\phi})), \mathbf{x}_i(\mathbf{p}))$
  end for
end if

Stage 3: Diffusion Model Training
Initialize diffusion model parameters $\boldsymbol{\nu}$
while not converged **do**
  Sample $t \sim \mathcal{U}(1, T)$, $\boldsymbol{\phi}_0 \sim \{\boldsymbol{\phi}_i\}_{i=1}^N$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
  Compute $\boldsymbol{\phi}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\phi}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$
  Update: $\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} - \nabla_{\boldsymbol{\nu}} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\nu}}(\boldsymbol{\phi}_t, t)\|^2$
end while
return $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$