Weight Space Representation Learning via Neural Field Adaptation

Supplementary Material

This supplementary material provides additional theoretical analysis, implementation details, and experimental results to support the main paper. Section 6 presents formal proofs demonstrating permutation symmetry in both additive and multiplicative LoRA parameterizations. Section 7 discusses related works in implicit neural representations. Section 8 provides comprehensive implementation details for the standalone MLP architecture, base model architecture and training, dataset preparation, and the complete training pipeline. Section 9 formally defines the evaluation metrics used throughout our experiments. Section 10 presents an ablation study examining the effectiveness of the hierarchical LoRA layer encoder. Section 11 presents weight space interpolation experiments. Section 12 provides additional qualitative generation results on FFHQ and ShapeNet datasets. Finally, Section 13 discusses limitations and future research directions.

6. Permutation Symmetry in LoRA

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

6.1. 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:

$$\mathbf{W}' = \mathbf{W} + \mathbf{BA} \tag{7}$$

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:

$$\mathbf{BAx} = \mathbf{B}(\mathbf{Ax}) \tag{8}$$

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:

$$\mathbf{BAx} = \mathbf{BP}^T\mathbf{PAx} = (\mathbf{BP}^T)(\mathbf{PA})\mathbf{x} \tag{9}$$

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

$$\mathbf{BA} = \tilde{\mathbf{B}}\tilde{\mathbf{A}} \tag{10}$$

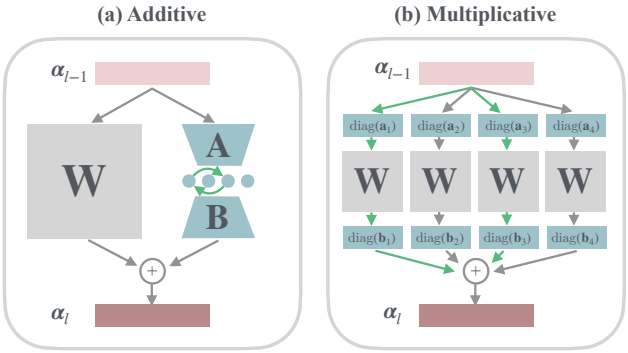

Figure 6. **Illustrating permutation symmetries within LoRA.** (a) Permutation symmetry in **additive** LoRA. Low rank matrices $\mathbf{A}$ and $\mathbf{B}$ could be seen as an encoder layer and a decoder layer. The order of the intermediate neurons could swapped without changing the output. (b) **multiplicative** 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.

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 [45]: 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 6(a) illustrates this symmetry, showing how the low rank matrices can be viewed as encoder and decoder layers where intermediate neurons can be reordered.

6.2. 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 3.2:

$$\mathbf{W}' = \mathbf{W} \odot \mathbf{BA} \tag{11}$$

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:

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_r \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_r^T \end{bmatrix} \tag{12}$$

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

Therefore:

$$\mathbf{W}' = \mathbf{W} \odot \left(\sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T \right) = \sum_{i=1}^{r} \mathbf{W} \odot \left(\mathbf{b}_i \mathbf{a}_i^T \right) \tag{13}$$

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:

$$\mathbf{W} \odot (\mathbf{b}_i \mathbf{a}_i^T) = \text{diag}(\mathbf{b}_i) \mathbf{W} \text{diag}(\mathbf{a}_i) \tag{14}$$

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

$$\begin{aligned} [\mathbf{W} \odot (\mathbf{b}_i \mathbf{a}_i^T)]_{jk} &= W_{jk} \cdot (b_i)_j \cdot (a_i)_k && (15) \\ &= (b_i)_j \cdot W_{jk} \cdot (a_i)_k && (16) \\ &= [\text{diag}(\mathbf{b}_i) \mathbf{W} \text{diag}(\mathbf{a}_i)]_{jk} && (17) \end{aligned}$$

Therefore:

$$\mathbf{W}' = \sum_{i=1}^{r} \text{diag}(\mathbf{b}_i) \mathbf{W} \text{diag}(\mathbf{a}_i) \tag{18}$$

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 2.1. *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 6(b) visualizes this symmetry by showing how multiplicative LoRA can be interpreted as parallel pathways that can be reordered without affecting the output.

Corollary 2.2. *Once permutation symmetry is eliminated, multiplicative LoRA weights are completely aligned with the channels in the base network.*

Proof. In Equation 18, 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. □

7. Related Works - Implicit Neural Representations

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 [10, 41], 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 [29], generative adversarial networks (GANs) [1, 3, 21] and shared layers [38]. The GAN-based works could be seen as extensions of the StyleGAN [20] 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 [7, 14] 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 [22] and transformers [5] to predict INR weights from input data via learned mappings as a way of data generation. Dupont *et al.* [8] 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

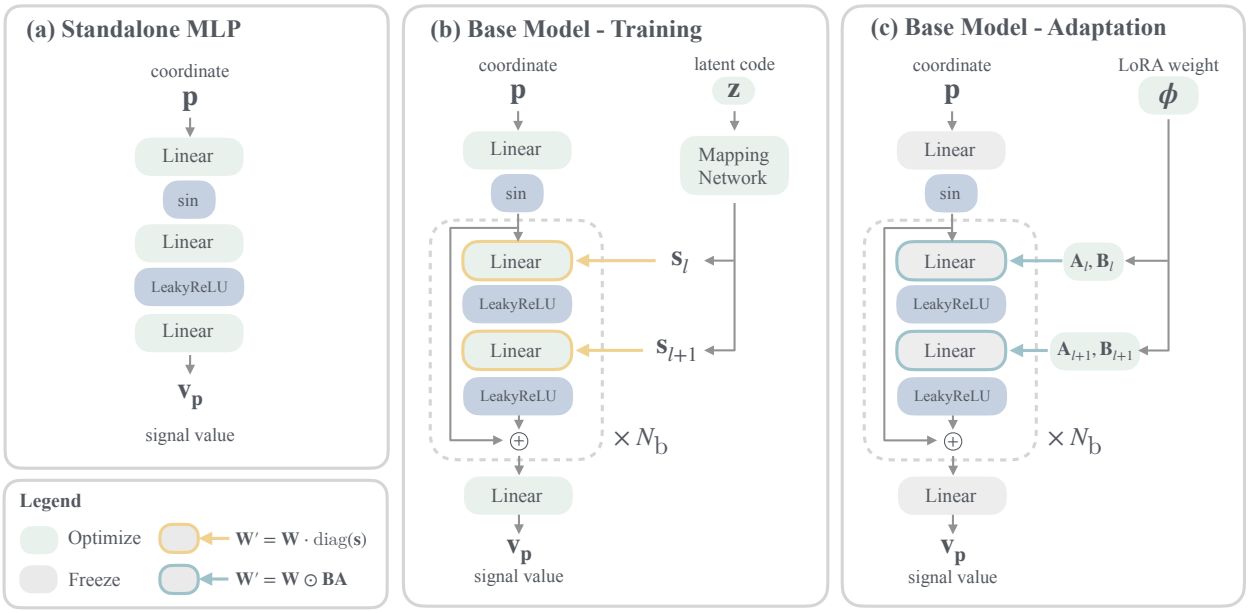

Figure 7. **Network architectures for weight space representations.** (a) Standalone MLP architecture with Fourier Feature layer followed by two linear layers. (b) Base model architecture with modulated fully connected layers. The network takes spatial coordinates $\mathbf{p}$ and style vector $\mathbf{s}$ 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.

as meaningful representations. HyperDiffusion [9] 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.

8. Implementation Details

8.1. Standalone MLP

As shown in Figure 7(a), The standalone MLP is a Fourier Feature [36] 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.* [9], 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.

8.2. Base Model Architecture

The base model follows the modulated neural field architecture widely adopted in generative neural field works [1, 3, 21], as illustrated in Figure 7(b). 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.

8.3. Base Model Training

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.

8.4. 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.

8.5. 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.

For the diffusion model architecture, we use a standard Diffusion Transformer (DiT) [31] for standalone MLP weights. For LoRA weights, we employ the hierarchical LoRA layer encoder described in Section 3.5 of the main paper. During training, we use the DDPM objective with a linear noise schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 2 \times 10^{-2}$ over $T = 500$ timesteps. For generation, we use DDIM sampling with 100 steps to efficiently sample new weight representations. 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.

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 $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$

9. 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.

9.1. 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 [16] or KID [2], as this allows for consistent evaluation across different data modalities. For 2D images, we use CLIP [33] as the feature extractor; for 3D shapes, we use a PointNet++ [32].

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) [12] 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 [2, 19]. 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 [2], 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 $\sigma_g = N_{\text{feature}}$.

9.2. Distance-based Metrics for 3D Shapes

For 3D shapes, we calculate distance-based metrics following [9, 26, 37, 47]. 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:

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

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.

10. Ablation Study

We examine the effectiveness of the hierarchical LoRA layer encoder introduced in Section 3.5. 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.

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**

11. Weight Space Interpolation

Figure 8 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 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 [37]. 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.

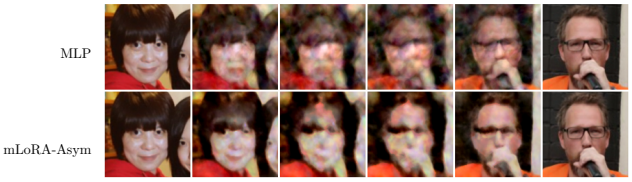

Figure 8. **Weight space interpolation.** Linear interpolation between pairs of mLoRA-Asym weight representations on FFHQ. Columns show the two endpoint instances (leftmost, rightmost) and intermediate interpolated reconstructions.

12. Additional Qualitative Results

We provide additional qualitative results on diffusion generation. Please see Figure 10 for results on ShapeNet Airplanes, Figure 11 for results on ShapeNet Multi, and Figure 12 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 9 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.

Figure 9. **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.

13. 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.

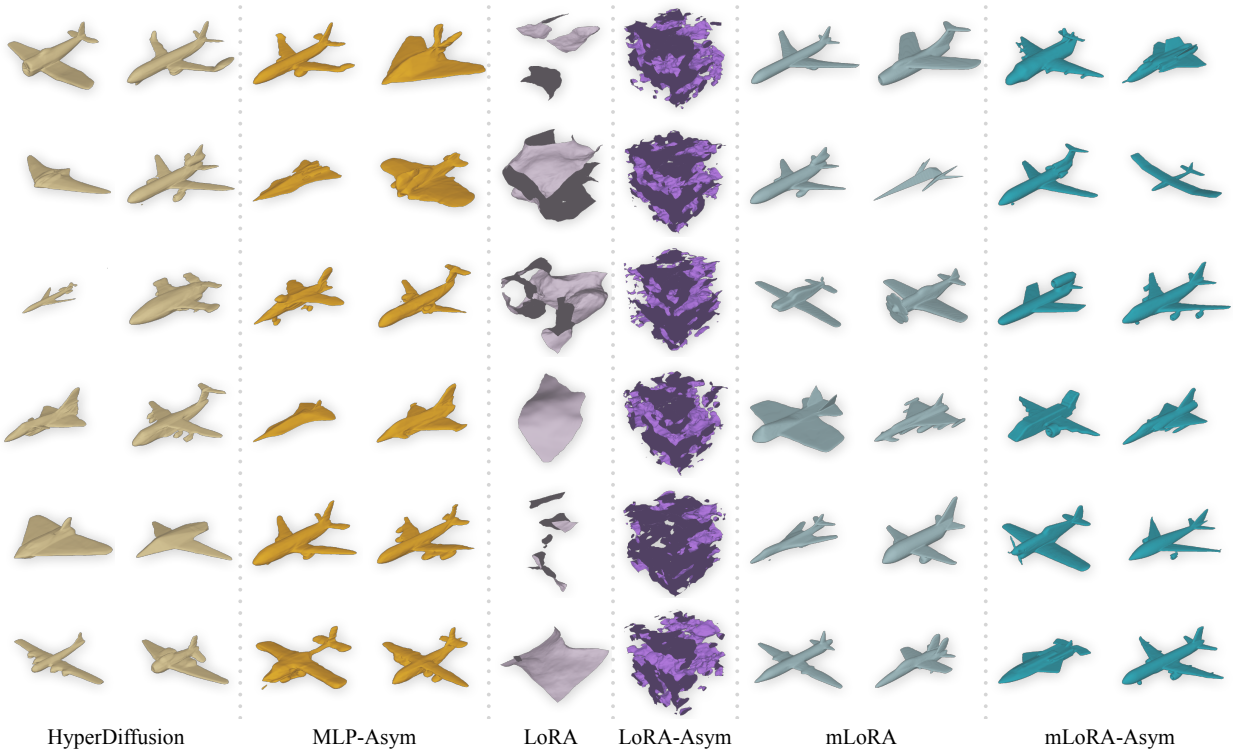

Figure 10. **Additional qualitative generation results on ShapeNet - Airplanes.**

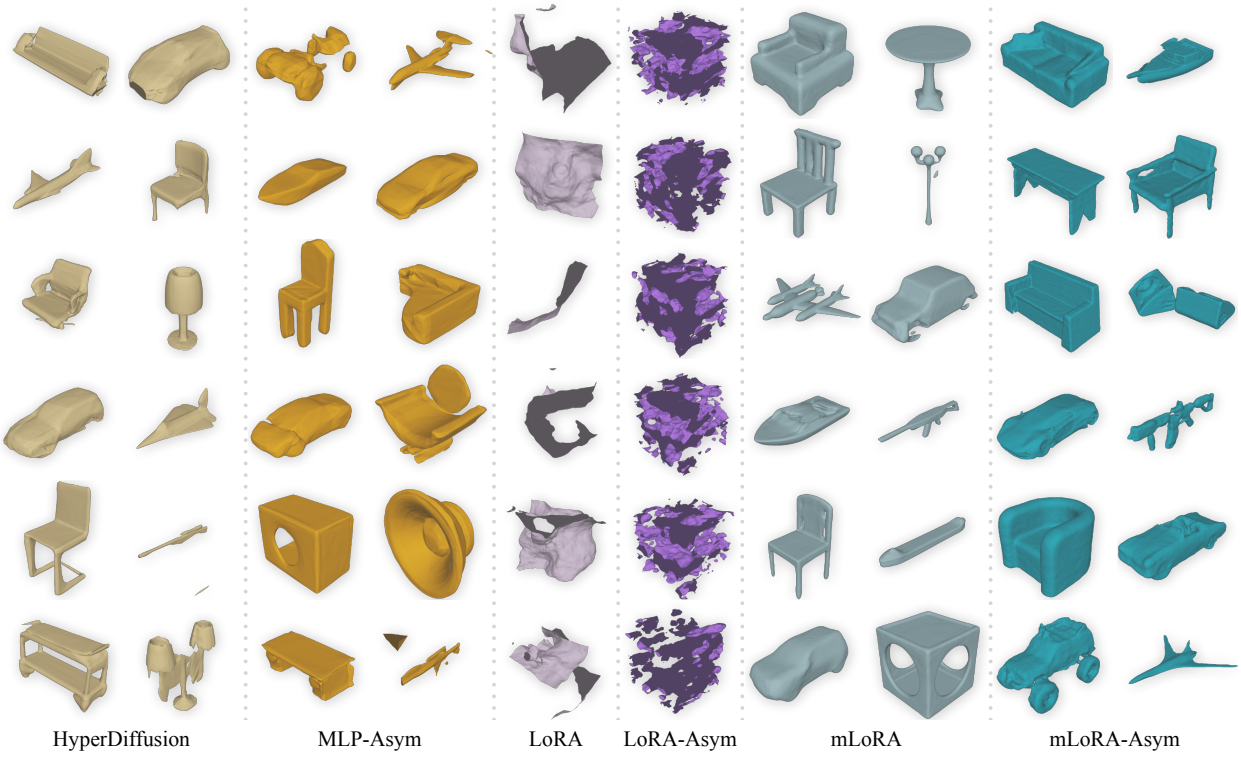

Figure 11. **Additional qualitative generation results on ShapeNet - Multi.**

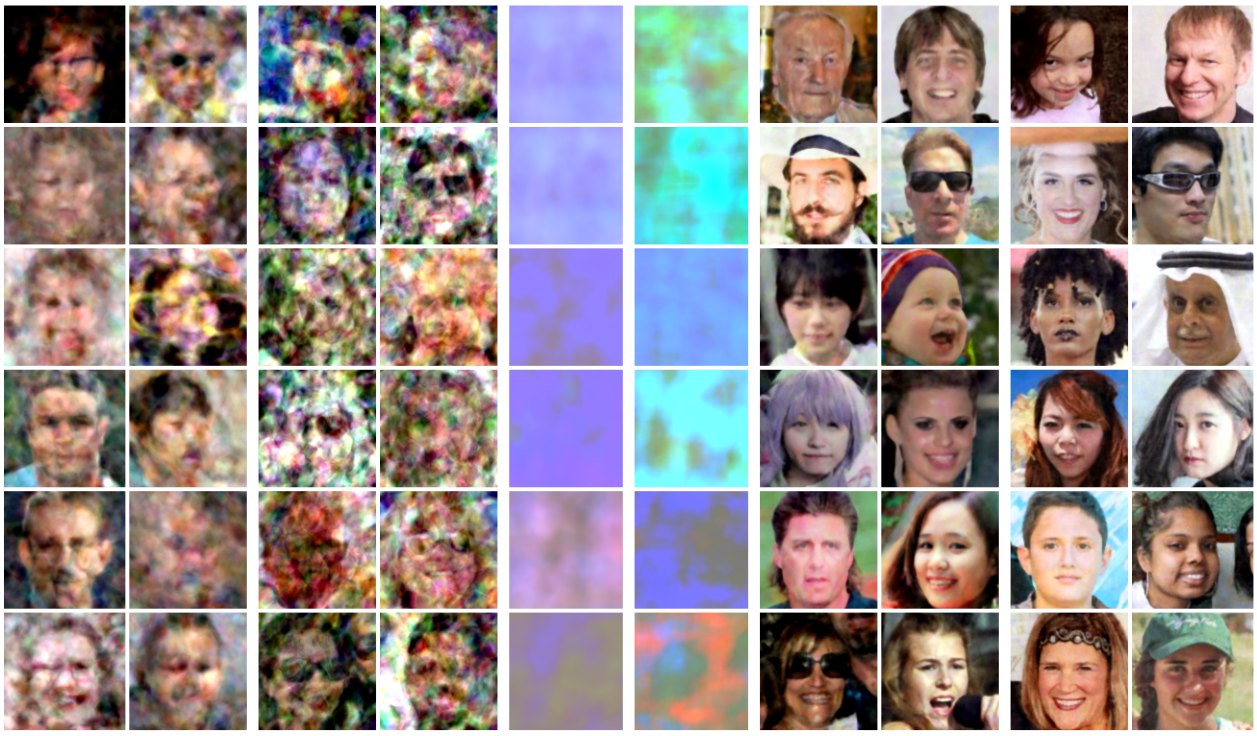

Figure 12. **Additional qualitative generation results on FFHQ.**

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14278–14287, 2021. 1, 2, 3, 5

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 6, 7, 5

[3] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. arxiv e-prints, page. *arXiv preprint arXiv:2012.00926*, 2020. 1, 2, 3, 5

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[5] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *European Conference on Computer Vision*, pages 170–187. Springer, 2022. 2

[6] Amil Dravid, Yossi Gandelsman, Kuan-Chieh Wang, Rameen Abdal, Gordon Wetzstein, Alexei Efros, and Kfir Aberman. Interpreting the weight space of customized diffusion models. *Advances in Neural Information Processing Systems*, 37:137334–137371, 2024. 2

[7] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 2

[8] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, 2022. 2

[9] Ziya Erkoç, Fangchang Ma, Cengiz Öztireli, and Pascal Fua. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *International Conference on Computer Vision*, 2023. 1, 2, 3, 4, 6, 7, 5

[10] Amer Essakine, Yanqi Cheng, Chun-Wun Cheng, Lipei Zhang, Zhongying Deng, Lei Zhu, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Where do we stand with implicit neural representations? a technical and performance survey. *arXiv preprint arXiv:2411.03688*, 2024. 2

[11] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020. 6

[12] Maurice Fréchet. Sur la distance de deux lois de probabilité. In *Annales de l'ISUP*, pages 183–198, 1957. 6, 5

[13] Y Gao and Others. Revisiting model merging: A statistical perspective. *arXiv preprint*, 2024. 2, 3

[14] Cameron Gordon, Lachlan E MacDonald, Hemanth Saratchandran, and Simon Lucey. D'oh: Decoder-only random hypernetworks for implicit neural representations. In *Proceedings of the Asian Conference on Computer Vision*, pages 2507–2526, 2024. 2

[15] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2016. 2

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6, 7, 5

[17] T Hospedales, A Antoniou, P Micaelli, and A Storkey. Meta-learning in neural networks: a survey. arxiv preprint arxiv: 200405439. 2020. 1

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1, 3

[19] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9307–9315, 2024. 5

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2, 5

[21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021. 1, 2, 3, 5

[22] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pages 496–510. Springer, 2019. 2

[23] M Kofinas, B Knyazev, Y Zhang, Y Chen, G J Burghouts, E Gavves, C G M Snoek, and D W Zhang. Graph neural networks for learning equivariant representations of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[24] D Lim, Y Gelberg, S Jegelka, H Maron, et al. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024. 1, 2, 4

[25] Derek Lim, Theo Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof. *Advances in Neural Information Processing Systems*, 37:28322–28358, 2024. 1, 4

[26] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021. 7, 5

[27] A Navon, A Shamsian, I Achituve, E Fetaya, G Chechik, and H Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pages 25790–25816, 2023. 2, 3

[28] O Ormaniec and Others. Fusion of graph convolutional networks via optimal transport. *arXiv preprint*, 2025. 2

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2, 3

[30] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2023. 1, 2, 4

[31] William S Peebles and Saining Xie. Scalable diffusion models with transformers. 2023 ieee. In *CVF International Conference on Computer Vision (ICCV)*, 2022. 4

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 7, 5

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 6, 7, 5

[34] C Singh and M Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, 2020. 2

[35] Jaisidh Singh, Diganta Misra, and Boris Knyazev6 Antonio Orvieto. Hyper-align: Efficient modality alignment via hypernetworks. In *Workshop on Neural Network Weights as a New Data Modality*. 1

[36] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020. 3

[37] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 7, 5, 6

[38] Kushal Vyas, Ahmed I Humayun, Aniket Dashpute, Richard G Baraniuk, Ashok Veeraraghavan, and Guha Balakrishnan. Learning transferable features for implicit neural representations. *Advances in Neural Information Processing Systems*, 37:42268–42291, 2024. 2

[39] K Wang and Others. Scaling weight space generative models. *arXiv preprint*, 2025. 2

[40] K Wang, Z Xu, Y Zhou, Z Zang, T Darrell, Z Liu, and Y You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024. 2

[41] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer graphics forum*, pages 641–676. Wiley Online Library, 2022. 1, 2

[42] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024. 1

[43] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022. 3, 6

[44] Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *arXiv preprint arXiv:2506.13018*, 2025. 1, 3

[45] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in neural information processing systems*, 36:24966–24992, 2023. 2, 5, 1

[46] Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. *Advances in neural information processing systems*, 36:77485–77502, 2023. 1, 2, 5

[47] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 7, 5