# Scaling Linear Mode Connectivity and Merging to Billion-Parameter Pretrained Transformers

**Tianyi Li**
MBZUAI
Tianyi.Li@mbzuai.ac.ae

**Zhiqiang Shen**
MBZUAI
Zhiqiang.Shen@mbzuai.ac.ae

## Abstract

Linear mode connectivity (LMC) provides a promising foundation for understanding and merging independently trained neural networks, but existing methods typically optimize the interpolation path from only one model endpoint, limiting their scalability and effectiveness for large pretrained transformers. We propose a novel and scalable framework for enabling LMC-based model merging for *billion-parameter pretrained Transformers*. Our method applies properly parameterized functionality-preserving weight transformations to align functionally equivalent solutions, and introduces a dual learning procedure in which both models jointly learn their corresponding transformations toward a shared linear interpolation path. This bidirectional optimization substantially reduces interpolation barriers and enables more reliable merging across large-scale architectures. Empirically, we show that our approach achieves near-zero loss barriers on WikiText for language models with medium-sized parameters, representing, to our knowledge, the first demonstration of near-barrier-free linear connectivity at this scale. In the vision domain, ViT-L maintains above 69% ImageNet top-1 accuracy throughout the interpolation path, while modern billion-parameter LLMs exhibit substantially reduced loss barriers. These results suggest that properly resolving parameter symmetries enables large pretrained transformers to be connected and merged through simple linear paths with substantially improved interpolation performance.

## 1 Introduction

Deep neural network weights have recently become an increasingly important object of study in the machine learning community [1–5]. Beyond specifying a model's input-output behavior, trained weights are increasingly viewed as reusable artifacts that can be inspected, edited, adapted, merged, or even treated as data for training other models [6, 7]. This perspective is especially important in the era of large pretrained models, where each checkpoint represents substantial investment in data, computation, optimization, and engineering [8]. Understanding the structure of weight space, and how different checkpoints are geometrically and functionally related, is therefore not only a theoretical question about loss landscapes, but also a practical foundation for scalable model reuse and composition.

A central idea for studying weight-space structure is linear mode connectivity (LMC) [2, 9]. Two models are linearly mode connected if the linear interpolation between their weights remains in a low-loss region. When such connectivity exists, simple weight interpolation becomes a natural and efficient mechanism for model merging, without requiring ensembles, additional architectures, or expensive retraining. However, independently trained models often fail to exhibit low-loss linear paths in their raw parameterization. Even when two networks implement similar functions, they may occupy apparently distant regions of weight space due to differences in initialization, training order, datasets, and hyperparameters [5, 10]. As a result, naive interpolation can encounter large loss barriers, making model merging unreliable.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

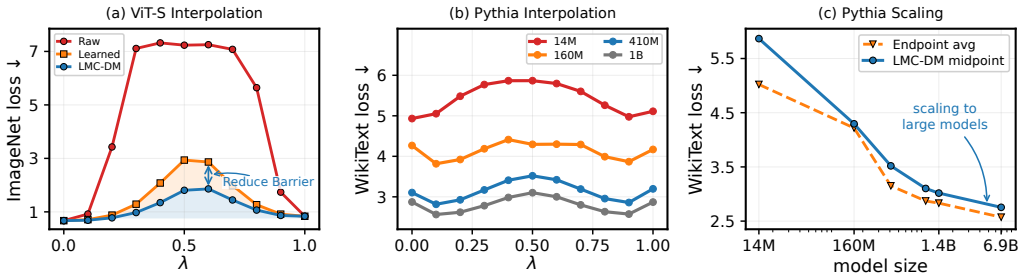


Figure 1: Loss barriers and scaling behavior. Left: Interpolation loss barriers for ViT-S under naive weight interpolation, learned matching, and LMC-DM (Ours). Middle: Interpolation loss curves for Pythia models across different parameter scales. Right: Scaling behavior of LMC-DM–merged Pythia models compared to endpoint models as model size increases.

A major reason for this failure is that neural networks exhibit many function-preserving symmetries: distinct parameter configurations can represent the same underlying function [11, 12]. A classic example is neuron permutation symmetry, where hidden units within a layer can be permuted together with corresponding inverse permutations in adjacent layers without changing the network function [13, 14, 12]. In small networks, resolving such symmetries can align independently trained models before interpolation and reveal low-loss paths that are hidden in the original coordinates [12, 15, 16]. Existing alignment methods commonly rely on activation matching, which uses data to align intermediate representations [17, 16, 3], or weight matching, which aligns parameters directly in a data-free manner [15, 3].

Beyond discrete symmetries like permutation, Transformers, however, introduce a substantially richer and more structured symmetry space than standard multilayer networks. Their residual connections, normalization layers, multi-head attention modules, positional encodings, and feed-forward blocks induce multiple classes of functionality-preserving transformations [18, 19]. Recent studies have begun to exploit these Transformer-specific symmetries to uncover hidden connectivity between trained models [20]. Nevertheless, most activation- or weight-matching methods remain task-agnostic: they align representations or parameters without directly optimizing the performance of interpolated models. This mismatch limits their ability to reduce the actual loss barrier, especially for pretrained Transformers where small misalignments can accumulate across many layers and severely degrade interpolation performance.

Learned matching partially addresses this issue by optimizing symmetry transformations using the task loss of interpolated models [19]. However, existing learned matching approaches still have several limitations. First, they typically exploit only a subset of the full Transformer symmetry group and often rely on simple parameterizations, such as unconstrained variables that are projected during the forward pass. Second, they apply symmetry transformations to only one endpoint model, leading to an asymmetric formulation in which one model is optimized toward a fixed counterpart. This one-sided optimization can create suboptimal landscapes and restrict the search for better-aligned representatives of both models. Consequently, prior demonstrations have largely been limited to models with millions or tens of millions of parameters trained on relatively small datasets. Whether LMC can be effectively revealed in large pretrained Transformers, and whether symmetry learning can scale to billion-parameter models, remains largely unexplored.

In this work, we propose a scalable framework for linear mode connectivity and model merging in billion-parameter pretrained Transformers. We systematically formulate a broad family of functionality-preserving weight transformations for Transformers, including normalization absorption, residual-space rotation and scaling, attention-head permutation, head-internal transformations with RoPE-aware variants, and feed-forward permutation and scaling. We parameterize these transformations under appropriate structural constraints and optimize them directly with respect to the loss along the interpolation path. Most importantly, we introduce a dual learning procedure, where both endpoint models learn their own symmetry transformations toward a shared linear path (Figure 2). This bidirectional formulation gives both models the flexibility to resolve their internal symmetries, substantially reduces interpolation barriers, and enables more reliable merging at scale. Empirically, we show that ViT-L maintains over 69% ImageNet-1K top-1 accuracy across the entire interpolation path; language models with 160M parameters achieve near-zero loss barriers on WikiText, the first such result in this setting, outperforming prior methods, including Weight Matching [3] and Learned Matching [19]; and modern billion-parameter LLMs exhibit substantially reduced loss

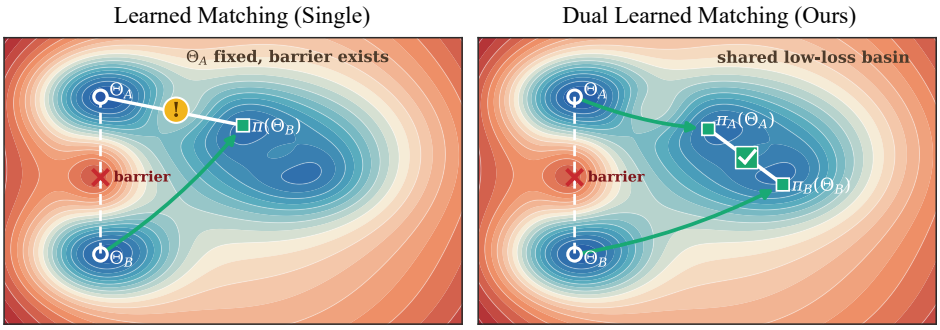


Figure 2: Schematic loss-landscape view of single and dual learned matching. Direct interpolation crosses a high-loss barrier. Single learned matching reduces but does not eliminate this barrier by optimizing one checkpoint toward a fixed reference. Dual learned matching jointly optimizes both checkpoints, yielding a lower barrier in a shared low-loss basin.

barriers (Figure 1). These results suggest that large pretrained Transformers can be effectively aligned and merged by simple linear interpolation when equipped with properly parameterized and jointly optimized symmetry transformations.

Our contributions are summarized as follows:

- We formulate a broad family of functionality-preserving Transformer symmetries. Specifically, our framework covers normalization absorption, residual-space rotation and scaling, attention-head permutation, head-internal transformations, and feed-forward permutation and scaling. We investigate practical parameterizations of continuous symmetry variables, enabling functionality-preserving transformations to be learned directly from the interpolation loss.
- We propose a dual learning procedure for bidirectional endpoint alignment. Unlike prior one-sided methods, both models learn their own transformations toward a shared linear interpolation path, leading to more effective optimization and lower interpolation barriers.
- We show strong and extensive empirical results that LMC scales to large pretrained Transformers. We demonstrate near-zero loss barriers for language models with 160M parameters on WikiText, over 69% ImageNet top-1 accuracy along the interpolation path for ViT-L, and only small loss barriers for billion-parameter LLMs.

## 2 Functionality-Preserving Symmetries in Transformers

This section introduces the functionality-preserving symmetry family used in our work. A symmetry is a reparameterization of the weights that changes the coordinates of internal representations while leaving the network function unchanged. Our goal is to expose equivalent parameterizations of two pretrained Transformers so that their weights can be better aligned before linear interpolation. In particular, we first convert normalization layers into parameter-free RMSNorm operators, which simplifies subsequent merging and enables global orthogonal symmetries on the residual stream. We then describe residual-space, attention-head, attention-circuit, and FFN symmetries. The highlighted symbols in the equations denote symmetry parameters.

### 2.1 Normalization and Residual-Space Symmetry

**Reparameterization of Normalization Layers.** Pretrained Transformers usually use normalization layers with learnable affine parameters, whereas the global residual-space symmetry below is stated for parameter-free RMSNorm. Previously, Ashkboos et al. [18] showed that LayerNorm and RMSNorm with parameters can be exactly converted into parameter-free RMSNorm by absorbing the affine parameters into adjacent linear layers. Given an RMSNorm layer with scaling gain $\boldsymbol{\gamma}$ followed by a linear map $\mathbf{W}$,

$$\mathrm{RMSNorm}_{\boldsymbol{\gamma}}(\mathbf{X})\mathbf{W} = \mathrm{RMSNorm}_0(\mathbf{X})\,\mathrm{Diag}(\boldsymbol{\gamma})\mathbf{W}, \tag{1}$$

where $\mathrm{RMSNorm}_0$ denotes RMSNorm without affine parameters. Thus the gain can be exactly absorbed into the following linear layer, allowing us to treat RMSNorm as parameter-free in the

symmetry analysis. For pretrained Transformers that use LayerNorm, the conversion requires an additional centering step. Let

$$\mathbf{C} = \mathbf{I} - \frac{1}{d}\mathbf{1}\mathbf{1}^\top \tag{2}$$

be the channel-centering operator. Writing $\mathrm{LayerNorm}_0$ for LayerNorm without affine parameters, we have

$$\mathrm{LayerNorm}_0(\mathbf{X}) = \mathrm{RMSNorm}_0(\mathbf{X}\mathbf{C}). \tag{3}$$

With affine parameters followed by a linear map,

$$\mathrm{LayerNorm}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{X})\mathbf{W} = \mathrm{RMSNorm}_0(\mathbf{X}\mathbf{C})\mathrm{Diag}(\boldsymbol{\gamma})\mathbf{W} + \boldsymbol{\beta}\mathbf{W}. \tag{4}$$

Thus the LayerNorm gain is absorbed into the following weight, and the LayerNorm bias is absorbed into the following bias term. By projecting residual writes onto the zero-mean subspace, a pre-LayerNorm Transformer can be represented as a parameter-free pre-RMSNorm Transformer on the centered residual stream. Appendix F gives the exact weight and bias transformations.

**Global Residual-Space Symmetry.** A Transformer using parameter-free RMSNorm has a global symmetry in the residual coordinate system. Under a row-vector convention, write one layer as

$$\mathbf{X}_{\ell+1} = \mathbf{X}_\ell + F_\ell(\mathrm{RMSNorm}(\mathbf{X}_\ell)). \tag{5}$$

Consider a scalar $g > 0$ and an orthogonal matrix $\mathbf{G}$, shared by all layers, that transform the residual stream as $\mathbf{X}'_\ell = \mathbf{X}_\ell g\mathbf{G}$. RMSNorm (assuming $\epsilon = 0$) removes the scalar but preserves the rotation:

$$\mathrm{RMSNorm}(\mathbf{X}\, g\mathbf{G}) = \mathrm{RMSNorm}(\mathbf{X})\, \mathbf{G}, \tag{6}$$

so each residual-read linear map can absorb the rotation by multiplying its input weight by $\mathbf{G}^\top$. To keep the residual stream in the transformed coordinates, every residual write, including attention output projections and FFN down projections, is also multiplied by $g\mathbf{G}$. Denote the block after these weight transformations by $F'_\ell$. It writes the original update in the new residual coordinates, giving

$$\mathbf{X}'_{\ell+1} = \mathbf{X}'_\ell + F'_\ell(\mathrm{RMSNorm}(\mathbf{X}'_\ell)) = (\mathbf{X}_\ell + F_\ell(\mathrm{RMSNorm}(\mathbf{X}_\ell)))\, g\mathbf{G} = \mathbf{X}_{\ell+1}\, g\mathbf{G}. \tag{7}$$

Therefore, $\mathbf{G}$ can be viewed as a global coordinate freedom of the residual stream.

### 2.2 Attention Head Permutations

Multi-head attention is invariant to permutations of its heads. For an input $\mathbf{X}$, write one attention layer as

$$\mathrm{MHA}(\mathbf{X}) = \sum_{i=1}^{h} \mathrm{softmax}\left(\frac{\mathbf{X}\mathbf{W}_{Q,i}(\mathbf{X}\mathbf{W}_{K,i})^\top}{\sqrt{d_h}}\right)\mathbf{X}\mathbf{W}_{V,i}\mathbf{W}_{O,i}. \tag{8}$$

For any head permutation $\pi$, let $\mathbf{P}_h = \mathbf{P} \otimes \mathbf{I}_{d_h}$ be the corresponding lifted permutation over concatenated head channels. In per-head notation, the reparameterization

$$\mathbf{W}'_{Q,i} = \mathbf{W}_{Q,\pi(i)}, \quad \mathbf{W}'_{K,i} = \mathbf{W}_{K,\pi(i)}, \quad \mathbf{W}'_{V,i} = \mathbf{W}_{V,\pi(i)}, \quad \mathbf{W}'_{O,i} = \mathbf{W}_{O,\pi(i)} \tag{9}$$

only reorders the summands, and therefore

$$\mathrm{MHA}'(\mathbf{X}) = \mathrm{MHA}(\mathbf{X}). \tag{10}$$

### 2.3 QK and OV Circuit Symmetries

Each attention head admits internal symmetries that preserve the attention logits and output.

For the QK circuit, multiplying the query projection by any invertible matrix can be compensated by multiplying the key projection by its inverse transpose. Concretely, for any invertible matrix $\mathbf{M}_{Q,i}$,

$$\mathbf{W}'_{Q,i} = \mathbf{W}_{Q,i}\mathbf{M}_{Q,i}, \qquad \mathbf{W}'_{K,i} = \mathbf{W}_{K,i}\mathbf{M}_{Q,i}^{-\top}, \tag{11}$$

which preserves the attention logits:

$$\mathbf{X}\mathbf{W}'_{Q,i}(\mathbf{X}\mathbf{W}'_{K,i})^\top = \mathbf{X}\mathbf{W}_{Q,i}(\mathbf{X}\mathbf{W}_{K,i})^\top. \tag{12}$$

For the OV circuit, multiplying the value projection by any invertible matrix can be compensated by multiplying the output projection by its inverse. For any invertible matrix $\mathbf{M}_{V,i}$,

$$\mathbf{W}'_{V,i} = \mathbf{W}_{V,i}\mathbf{M}_{V,i}, \qquad \mathbf{W}'_{O,i} = \mathbf{M}_{V,i}^{-1}\mathbf{W}_{O,i}, \tag{13}$$

which preserves the value-output product:

$$\mathbf{X}\mathbf{W}'_{V,i}\mathbf{W}'_{O,i} = \mathbf{X}\mathbf{W}_{V,i}\mathbf{W}_{O,i}. \tag{14}$$

In practice, additional architectural constraints (e.g., positional encodings such as RoPE [21]) can further restrict these symmetries.

**RoPE-compatible QK symmetry.** With rotary position embeddings (RoPE) [21], the attention logit between positions $m$ and $n$ is

$$\mathbf{x}_m\mathbf{W}_{Q,i}\mathbf{R}_{m-n}\mathbf{W}_{K,i}^{\top}\mathbf{x}_n^{\top}, \qquad \mathbf{R}_{m-n} := \mathbf{R}_m\mathbf{R}_n^{\top}, \tag{15}$$

where $\mathbf{R}_t$ applies a two-dimensional rotation to each RoPE frequency pair. Under Eq. (11), preserving this logit requires $\mathbf{M}_{Q,i}\mathbf{R}_t = \mathbf{R}_t\mathbf{M}_{Q,i}$. An orthogonal RoPE-compatible QK symmetry is given by independent rotations within the RoPE frequency pairs:

$$\mathbf{M}_{Q,i} = \mathrm{blockdiag}\big(\mathbf{R}(\phi_{i,1}), \ldots, \mathbf{R}(\phi_{i,d_h/2})\big). \tag{16}$$

This restricted QK transformation preserves the attention logits in the presence of RoPE.

### 2.4 FFN Permutation

A standard two-layer FFN admits hidden-channel permutations. For

$$\mathrm{FFN}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{17}$$

any hidden permutation $\mathbf{P}_f$ yields

$$\mathbf{W}'_1 = \mathbf{W}_1\mathbf{P}_f, \qquad \mathbf{b}'_1 = \mathbf{b}_1\mathbf{P}_f, \qquad \mathbf{W}'_2 = \mathbf{P}_f^{\top}\mathbf{W}_2, \tag{18}$$

and therefore

$$\mathrm{FFN}'(\mathbf{X}) = \mathrm{FFN}(\mathbf{X}). \tag{19}$$

**GLU rescaling symmetry.** Beyond the hidden-channel permutation above, GLU-style FFNs have an additional value-branch rescaling symmetry: scaling the value branch can be canceled exactly in the down projection, while the gate branch is unchanged.

$$\mathrm{GLU}(\mathbf{X}) = \left(\phi(\mathbf{X}\mathbf{W}_{\mathrm{gate}}) \odot \mathbf{X}\mathbf{W}_{\mathrm{up}}\right)\mathbf{W}_{\mathrm{down}}, \tag{20}$$

For any invertible diagonal matrix $\mathbf{S}$, the reparameterization

$$\mathbf{W}'_{\mathrm{up}} = \mathbf{W}_{\mathrm{up}}\mathbf{S}, \qquad \mathbf{W}'_{\mathrm{down}} = \mathbf{S}^{-1}\mathbf{W}_{\mathrm{down}} \tag{21}$$

satisfies

$$\mathrm{GLU}'(\mathbf{X}) = \mathrm{GLU}(\mathbf{X}). \tag{22}$$

## 3 Matching Algorithms

Given the functionality-preserving symmetries introduced in Section 2, we now study how to obtain proper symmetries to align two independently trained models. We consider a design space of matching approaches, including weight matching and learned matching, and discuss how to parameterize these symmetries to make them learnable through gradient descent. Building on this, we introduce **dual learned matching**, which assigns learnable transformations to both checkpoints, yielding a more favorable optimization landscape for finding low-barrier linear paths.

### 3.1 From Weight Matching to Learned Matching

**Weight Matching.** Introduced by Ainsworth et al. [3], weight matching formulates the weight alignment problem as maximizing weight similarity under the symmetry family. The original paper only discusses permutation symmetries in MLP networks, but the same procedure can be extended to more complex symmetries in Transformers. Following the adoption by Theus et al. [19], we first recover the global residual-space symmetry:

$$\mathbf{G} = \arg\min_{\mathbf{G}^\top\mathbf{G}=\mathbf{I}} \left\|\mathbf{R}^A - \mathbf{R}^B\mathbf{G}\right\|_F^2, \qquad g = \frac{\langle \mathbf{R}^B\mathbf{G}, \mathbf{R}^A\rangle}{\|\mathbf{R}^B\mathbf{G}\|_F^2}. \tag{23}$$

Here $\mathbf{R}^A$ and $\mathbf{R}^B$ are the weights along the residual path collected from both models; they may include the token embedding and all residual read/write matrices. The closed-form solution is given by the SVD of $\mathbf{R}^{B\top}\mathbf{R}^A$. Then, for each layer $\ell$, we recover the head permutation by comparing the induced QK and OV circuits. For each head $i$ from model $A$ and head $j$ from model $B$ we define

$$\mathbf{C}_{ij}^{\text{head}} = \left\|\mathbf{W}_{Q,i}^A\mathbf{W}_{K,i}^{A\top} - \mathbf{G}^\top\mathbf{W}_{Q,j}^B\mathbf{W}_{K,j}^{B\top}\mathbf{G}\right\|_F^2 + \left\|\mathbf{W}_{V,i}^A\mathbf{W}_{O,i}^A - \mathbf{G}^\top\mathbf{W}_{V,j}^B\mathbf{W}_{O,j}^B\mathbf{G}\right\|_F^2, \tag{24}$$

$$\mathbf{P}_{h,\ell} = \arg\min_{\mathbf{P}\in\Pi_h} \sum_{i,j} P_{ij}\,\mathbf{C}_{ij}^{\text{head}}. \tag{25}$$

This compares heads through their functional QK and OV circuits rather than their raw weights, and therefore avoids the ambiguity from per-head internal invertible symmetries. After the head permutation is fixed, the remaining QK and OV symmetries inside each head can be solved independently in closed form, so we omit their explicit formulas here.

For FFN layers, we solve the corresponding bilinear assignment problem:

$$\mathbf{P}_{f,\ell} = \arg\max_{\mathbf{P}\in\Pi_f} \left\langle \mathbf{W}_{1,\ell}^A, \mathbf{G}^\top\mathbf{W}_{1,\ell}^B\mathbf{P}\right\rangle_F + \left\langle \mathbf{W}_{2,\ell}^A, \mathbf{P}^\top\mathbf{W}_{2,\ell}^B\mathbf{G}\right\rangle_F. \tag{26}$$

The same construction extends directly to gated FFNs. Since $\mathbf{G}$, head permutations, and FFN permutations are coupled, these matching steps can be iterated, as in the SOBLAP-style procedure by Ainsworth et al. [3]. After weight matching, the two models are in a substantially better common coordinate system, but weight matching is loss-agnostic and can therefore still leave a visible loss barrier along the linear path.

**Learned Matching.** Following Theus et al. [19], learned matching refines the weight matching solution by directly optimizing the interpolation loss. Starting from the matched model $\pi(\Theta_B)$, we can do linear interpolation with the other model $\Theta_A$:

$$\Theta_{\text{interp}} = \lambda\,\Theta_A + (1-\lambda)\,\pi(\Theta_B), \qquad \lambda \in [0,1]. \tag{27}$$

We then optimize the symmetry parameters in $\pi$ to minimize the interpolation loss through gradient descent. In this way, learned matching uses task loss to refine the loss-agnostic alignment from weight matching into a lower-barrier linear path.

### 3.2 Parameterizing Symmetries

To optimize continuous symmetry transformations using gradient-based methods, we consider a range of parameterizations for orthogonal and invertible symmetries, differing in how they handle structural constraints during training, as summarized in Table 1.

Table 1: Continuous symmetry parameterizations.

| Method | Form | Covers |
|---|---|---|
| **Cayley** | $\mathbf{Q} = (\mathbf{I}-\mathbf{A})^{-1}(\mathbf{I}+\mathbf{A})$, $\mathbf{A} = \mathbf{S} - \mathbf{S}^\top$ | $SO(N)\backslash\{-1\}$ |
| **MatExp (orth.)** | $\mathbf{Q} = \exp(\mathbf{A})$ | $SO(N)$ |
| **MatExp (inv.)** | $\mathbf{M} = \exp(\mathbf{U})$ | Subset of $\mathrm{GL}^+(N)$ |
| **Polar** | $\mathbf{M} = \mathbf{Q}(\mathbf{L}\mathbf{L}^\top)$, $\mathbf{Q} = \mathrm{Cayley}(\mathbf{A})$, $\mathbf{L}$ lower-tri., diag$> 0$ | Cayley-limited subset of $\mathrm{GL}^+(N)$ |

**Orthogonal.** For orthogonal symmetries, we primarily consider the Cayley transform and the matrix exponential. The Cayley transform is computationally efficient, numerically stable, and straightforward to implement, but it cannot represent orthogonal matrices whose spectrum contains the eigenvalue $-1$, while the matrix exponential provides a standard parameterization of $SO(N)$ with a higher computational cost. We avoid Householder products due to their relative inefficiency on

GPUs, as well as unconstrained parameterizations followed by projection, which exhibited instability during optimization in our experiments.

**Invertible.** For invertible symmetries, we explore the matrix exponential and a polar-decomposition-based parameterization, referred to as polar. The former guarantees invertibility and a positive determinant by setting $\mathbf{M} = \exp(\mathbf{U})$, although it only covers the subset of $\mathrm{GL}^+(N)$ that admits a real matrix logarithm. The latter decomposes the transform into a Cayley-parameterized orthogonal factor and a Cholesky-parameterized symmetric positive-definite factor. Additionally, we evaluate unconstrained direct parameterization as a baseline for invertible transforms.

### 3.3 Dual Learned Matching

In both weight matching and learned matching, only one of the two checkpoints is transformed to align with the other, a design choice motivated by simplicity and efficiency. However, this one-sided procedure can be suboptimal, as it requires aligning to a fixed target in the original coordinate system, which may lead to a less favorable optimization landscape. A more natural parameterization is to assign learnable transformations to both checkpoints and optimize them jointly, allowing the models to meet in a shared coordinate system.

We call this approach **Dual Learned Matching**, and it can be formulated as:

$$\Theta_{\mathrm{interp}} = \lambda\, \pi_A(\Theta_A) + (1-\lambda)\, \pi_B(\Theta_B), \qquad \lambda \in [0,1]. \tag{28}$$

Starting from the weight matching solution, we optimize the transformations on both checkpoints directly through interpolation loss, as shown in Algorithm 1. In practice, we optimize the continuous symmetries on both sides, so that both models can move toward a shared coordinate system rather than forcing one model to match the other exactly. For discrete FFN and attention-head permutations, we find that optimizing them following Learned Matching [19] changes very few assignments relative to the weight-matching solution and has little effect on the loss barrier, but incurs substantial computational overhead for large models. We therefore keep them fixed at the weight-matching solution.

**Algorithm 1** Dual learned matching (LMC-DM)

**Require:** Checkpoints $\Theta_A, \Theta_B$; dataset $\mathcal{D}$; sampling interval $[a, b]$; iterations $T$; learning rate $\eta$.

Initialize $\Pi_A, \Pi_B$ from weight matching.
$\Pi_m = (g_m, \mathbf{G}_m, \mathbf{M}_{QK,m}, \mathbf{M}_{OV,m}), \quad m \in \{A, B\}$.
$\mathbf{G}_m, \mathbf{M}_{QK,m}, \mathbf{M}_{OV,m} \in \{\textsc{Cayley}, \textsc{Polar}, \textsc{Exp}\}$.
**for** $t = 1$ **to** $T$ **do**
  $\Theta'_A \leftarrow \pi_{\Pi_A}(\Theta_A), \quad \Theta'_B \leftarrow \pi_{\Pi_B}(\Theta_B)$.
  Sample $\lambda \sim \mathcal{U}(a, b)$ and a minibatch $B \sim \mathcal{D}$.
  Interpolate: $\Theta_{\mathrm{interp}} \leftarrow \lambda\, \Theta'_A + (1-\lambda)\, \Theta'_B$.
  Compute $\mathcal{J} \leftarrow \frac{1}{|B|} \sum_{(x,y)\in B} \mathcal{L}(\Theta_{\mathrm{interp}}; x, y)$.
  Update $\Pi_A, \Pi_B$ by gradient descent on $\mathcal{J}$.
**end for**
**return** $\Pi_A, \Pi_B$.

## 4 Experiments

### 4.1 Experimental Setup

We use publicly available independently trained checkpoints with identical architectures, including Vision Transformers [22] and language models [23, 24], to evaluate our method. We assess linear mode connectivity by sampling points along the linear interpolation between two endpoints and measuring the loss along the path. We report the loss barrier [12, 19]:

$$\max_{\lambda\in[0,1]} \Big[\mathcal{L}(\Theta_\lambda) - \big(\lambda \mathcal{L}(\Theta_A) + (1-\lambda)\mathcal{L}(\Theta_B)\big)\Big] \tag{29}$$

approximated over the sampled path. For ViTs, we also evaluate ImageNet-1K [25] top-1 accuracy along the interpolation path, using interpolation intervals of 0.1. For language models, we report WikiText [26] perplexity additionally. Weight Matching (WM) and Learned Matching (LM) are methods from Ainsworth et al. [3] and Theus et al. [19], respectively. LMC-DM denotes our dual learned matching procedure with learnable parameterization of continuous symmetries. Enhanced learned matching refers to the same procedure without dual parameterization. We use Cayley parameterization for orthogonal symmetries and polar parameterization for invertible symmetries by default for our methods. We consider two initialization schemes based on weight matching (WM): an

Table 2: Loss barrier comparison across ViT and Pythia models. Lower is better.

| Method | ViT (ImageNet-1K) | | | Pythia (WikiText) | | | |
|---|---|---|---|---|---|---|---|
| | Small | Base | Large | 14M | 160M | 410M | 1B |
| Raw Interpolation | 6.58 | 6.57 | 6.47 | 9.89 | 6.28 | 4.99 | 4.60 |
| Weight Matching [3] | 6.58 | 6.50 | 6.48 | 8.17 | 6.04 | 5.65 | 4.75 |
| Learned Matching [19] | 2.18 | 2.00 | 1.23 | 1.26 | 0.39 | 0.57 | 0.42 |
| Enhanced Learned Matching (Ours) | 1.20 | 1.05 | 0.71 | 1.00 | 0.27 | 0.44 | 0.28 |
| **Dual Learned Matching (Ours)** | **1.09** | **0.82** | **0.66** | **0.86** | **0.18** | **0.37** | **0.23** |

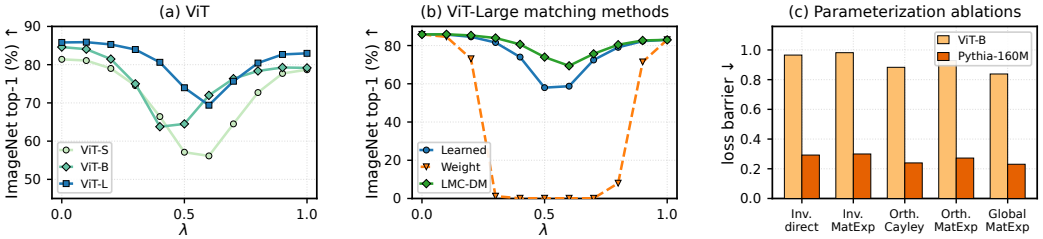


Figure 3: Summary of interpolation and parameterization results. (a) LMC-DM across ViT-Small, ViT-Base, and ViT-Large. (b) ViT-Large accuracy paths under learned matching, weight matching, and LMC-DM. (c) Loss-barrier comparison across symmetry parameterizations on ViT-Base and Pythia-160M.

absorbed initialization, where the WM solution is absorbed into the endpoints and the symmetries are initialized as identity, and a non-absorbed initialization, where the symmetries are initialized with the WM solution. The two schemes are equivalent at initialization in terms of the resulting interpolation. Additional experimental details are provided in Appendix D.

### 4.2 Main Results

Table 2 summarizes the loss-barrier comparison across ViT and Pythia scales. Simple WM performs poorly on modern vision and language models. While LM substantially reduces the barrier compared to WM, it still leaves a significant gap to zero-barrier connectivity. Our enhanced version of learned matching further reduces the barrier. LMC-DM achieves the lowest barrier among all methods, indicating that jointly optimizing both endpoints yields better interpolation paths than one-sided learned matching. Figure 3 shows the ViT accuracy curves during interpolation.

### 4.3 Ablation Studies

**Symmetry Components.** We first ablate the symmetry components used in LMC-DM, as shown in Table 4. Disabling the QK or OV symmetries leads to a significantly higher barrier, especially for ViTs. For Pythia, disabling QK symmetries has a smaller impact. Disabling both symmetries further increases the barrier, but still performs better than standard learned matching, which does not leverage attention symmetries and uses a single-sided parameterization. The global scaling symmetry has a smaller but non-negligible effect.

**Symmetry Parameterization.** We then compare different parameterizations for continuous symmetries in Figure 3. Despite having fewer degrees of freedom, Cayley-based orthogonal parameterization outperforms unconstrained and matrix-exponential parameterizations for invertible symmetries. This may be because the Cayley transform is more numerically stable and easier to optimize than the matrix exponential. Combined with Cayley, the polar parameterization for invertible symmetries achieves the best performance, balancing both stability and expressivity.

Table 3: Single vs. dual endpoint matching.

| Setting | ViT-B ↓ | Pythia-160M ↓ |
|---|---|---|
| Single *w. Absorb* | 1.05 | 0.27 |
| Single *w/o Absorb* | 1.47 | 0.41 |
| Dual *w/o Absorb* | 0.82 | 0.21 |
| Dual *w. Absorb* | **0.82** | **0.18** |

**Single vs. Dual Matching.** We compare whether the matching transformation should be learned on one endpoint or on both endpoints in Table 3. Dual matching gives both checkpoints room to move

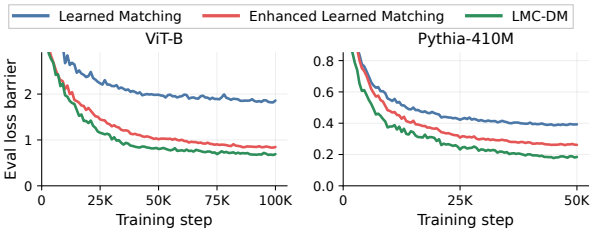


**Figure 4: Evaluation Loss Barriers.** We track ViT-B and Pythia-410M during matching. LMC-DM reduces barriers faster and reaches lower final barriers than learned matching and the enhanced learned matching. For efficiency, these evaluation barriers are measured on a subset rather than the full test or validation set.

Table 4: Ablation study on ViT-Base and Pythia-160M. Small lower-right numbers report the raw numerical change from the full LMC-DM baseline; green indicates better and red indicates worse. Ours (default) uses dual matching, Cayley parameterization for the global transformation, and polar parameterization for attention transformations.

| Ablation | ViT-Base | | | Pythia-160M | | |
|---|---|---|---|---|---|---|
| | Barrier ↓ | Worst Acc ↑ | Mid. Acc ↑ | Barrier ↓ | Worst PPL ↓ | Mid. PPL ↓ |
| Ours *(default)* | 0.82 | 63.77 | 64.51 | 0.18 | 82.47 | 73.33 |
| *Component removals* | | | | | | |
| w/o. QK transform | 0.97 $_{+0.15}$ | 60.68 $_{-3.09}$ | 61.80 $_{-2.71}$ | 0.18 $_{+0.00}$ | 81.81 $_{-0.66}$ | 76.85 $_{+3.52}$ |
| w/o. OV transform | 1.17 $_{+0.35}$ | 56.85 $_{-6.92}$ | 61.04 $_{-3.47}$ | 0.37 $_{+0.19}$ | 97.03 $_{+14.56}$ | 82.97 $_{+9.64}$ |
| w/o. QK and OV | 1.48 $_{+0.66}$ | 50.98 $_{-12.79}$ | 54.96 $_{-9.55}$ | 0.37 $_{+0.19}$ | 97.06 $_{+14.59}$ | 82.11 $_{+8.78}$ |
| w/o. global scale | 0.82 $_{+0.00}$ | 63.50 $_{-0.27}$ | 65.39 $_{+0.88}$ | 0.23 $_{+0.05}$ | 86.01 $_{+3.54}$ | 77.53 $_{+4.20}$ |
| *Attention parameterization* | | | | | | |
| Invertible, direct | 0.97 $_{+0.15}$ | 60.72 $_{-3.05}$ | 63.68 $_{-0.83}$ | 0.29 $_{+0.11}$ | 90.20 $_{+7.73}$ | 80.31 $_{+6.98}$ |
| Invertible, MatExp | 0.98 $_{+0.16}$ | 60.26 $_{-3.51}$ | 63.57 $_{-0.94}$ | 0.30 $_{+0.12}$ | 90.59 $_{+8.12}$ | 78.06 $_{+4.73}$ |
| Orthogonal, Cayley | 0.88 $_{+0.06}$ | 62.42 $_{-1.35}$ | 63.62 $_{-0.89}$ | 0.24 $_{+0.06}$ | 85.18 $_{+2.71}$ | 71.86 $_{-1.47}$ |
| Orthogonal, MatExp | 0.93 $_{+0.11}$ | 61.53 $_{-2.24}$ | 63.26 $_{-1.25}$ | 0.27 $_{+0.09}$ | 88.10 $_{+5.63}$ | 73.99 $_{+0.66}$ |
| *Global parameterization* | | | | | | |
| MatExp | 0.84 $_{+0.02}$ | 63.33 $_{-0.44}$ | 65.63 $_{+1.12}$ | 0.23 $_{+0.05}$ | 86.00 $_{+3.53}$ | 76.45 $_{+3.12}$ |

toward a shared basin, which consistently lowers the barrier compared with the single-sided variants. Absorbing the weight-matching solution into the endpoints leverages clean identical initialization for symmetries, which further reduces the barrier. Figure 4 further shows the convergence of evaluation loss barriers during matching.

### 4.4 Scaling Up to Larger Models

We apply our method to Pythia and OLMo models up to the billion-parameter scale (Table 6 in Appendix D). As model size increases, the loss barrier for Pythia remains low, even at 6.9B, while OLMo-7B exhibits a significantly higher barrier. These results indicate that large pretrained Transformers can still exhibit linear mode connectivity. At the same time, the discrepancy between Pythia and OLMo suggests that LMC in large models is more nuanced and influenced by factors beyond scale, such as training data, initialization, and optimization dynamics.

## 5 Related Work

**Linear Mode Connectivity.** Early work [2, 9] introduced mode connectivity by showing that independently trained networks can be connected through stable low-loss paths. Follow-up studies [16, 12] showed that high linear interpolation barriers often arise from unresolved permutation symmetries, and that resolving them can largely recover linear mode connectivity (LMC). Frankle et al. [5] further connected LMC to the lottery ticket hypothesis, showing that lottery tickets can lie in a shared linearly connected basin. More recent work extends LMC to layer-wise connectivity, broader symmetry groups, multi-model settings, and modern architectures such as Transformers and MoEs [27, 28, 19, 29, 30].

**Weight Symmetries and Invariances.** This refers to the family of transformations of model parameters that exactly preserve the functionality. Neuron permutation symmetries in MLPs have long been known [13, 14]. Ashkboos et al. [18] observed orthogonal symmetries in the residual space of Transformers, along with absorbable symmetries between normalization and linear layers. Zhang

et al. [20] and Theus et al. [19] identified head-wise symmetries in Transformers. A systematic survey of parameter symmetries is provided by Zhao et al. [4].

**Pretrained Model Merging.** Pretrained model merging aims to combine multiple independently trained checkpoints into a single model, and is significantly more challenging than task arithmetic in the fine-tuning regime, as independently trained models often diverge into different basins [5, 31, 32]. OT-Fusion [15] first leveraged optimal transport to align neurons, enabling data-driven model merging that outperforms naive interpolation. Subsequent work, such as Git Re-Basin [3], proposed practical approaches to align models under permutation symmetries to achieve low-barrier fusion. More recent studies have extended these ideas to pretrained Transformers [33, 20, 19].

## 6 Conclusion

We studied whether and to what extent linear mode connectivity can be achieved in modern pretrained Transformers by accounting for their weight-space symmetries. We explicitly parameterize the relevant symmetries in modern Transformers and optimize them with dual learned matching, which allows both endpoints to be symmetrically aligned toward a shared linear interpolation path. This alignment places independently trained checkpoints in a shared coordinate system, enabling meaningful linear interpolation across ViTs and LLMs up to the billion-parameter scale. More broadly, this perspective suggests that modern large models are not isolated endpoints in weight space, but rather symmetry-related representations that can be aligned and connected. Future work includes extending this framework to broader architectures and training settings, applying it to practical model merging and ensembling, and further interpreting the geometry of the loss landscape under these symmetries. Limitations and further discussion can be found in Appendix A and Appendix H.

## Acknowledgments and Disclosure of Funding

This work was supported by the MBZUAI-WIS Joint Program for Artificial Intelligence Research. The authors declare no competing interests.

## References

[1] Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, Rose Yu, Haggai Maron, Yanfang Ye, Lu Yin, et al. A survey of weight space learning: Understanding, representation, and generation. *arXiv preprint arXiv:2603.10090*, 2026.

[2] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

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

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

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

[7] Boya Zeng, Yida Yin, Zhiqiu Xu, and Zhuang Liu. Generative modeling of weights: Generalization or memorization? *arXiv preprint arXiv:2506.07998*, 2025.

[8] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[9] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[10] Devin Kwok, Gül Sena Altıntaş, Colin Raffel, and David Rolnick. The butterfly effect: Neural network training trajectories are highly sensitive to initial conditions. *arXiv preprint arXiv:2506.13234*, 2025.

[11] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[12] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

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

[14] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[15] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[16] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[17] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pages 196–212. PMLR, 2015.

[18] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

[19] Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. *arXiv preprint arXiv:2506.22712*, 2025.

[20] Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion. *arXiv preprint arXiv:2502.00264*, 2025.

[21] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[23] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pages 2397–2430. PMLR, 2023.

[24] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander,

Dustin Schwenk, Saurabh Shah, Will Smith, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[26] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[27] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity. *Advances in neural information processing systems*, 36:60853–60877, 2023.

[28] Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layer-wise linear mode connectivity. *arXiv preprint arXiv:2307.06966*, 2023.

[29] Akira Ito, Masanori Yamada, and Atsutoshi Kumagai. Linear mode connectivity between multiple models modulo permutation symmetries. In *Forty-second International Conference on Machine Learning*, 2025.

[30] Viet-Hoang Tran, Van Hoan Trinh, Khanh Vinh Bui, and Tan M Nguyen. On linear mode connectivity of mixture-of-experts architectures. *arXiv preprint arXiv:2509.11348*, 2025.

[31] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[32] 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. *ACM Computing Surveys*, 58(8):1–41, 2026.

[33] Neha Verma and Maha Elbayad. Merging text transformer models from different initializations. *arXiv preprint arXiv:2403.00986*, 2024.

[34] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[35] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu`.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[37] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[38] Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, 2017.

[39] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[40] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

# Appendix

## A Limitations

Our framework relies on explicitly parameterized functionality-preserving symmetries, and therefore may not capture all sources of misalignment between independently pretrained models. The current formulation mainly focuses on models with compatible architectures, dimensions, and tokenization spaces, extending it to heterogeneous architectures or models trained with substantially different objectives remains challenging. Besides, in some Transformer architectures, the available functionality-preserving symmetries are structurally limited. In such cases, the effectiveness of our method may be limited due to the reduced symmetry capacity.

## B Societal Impacts

This work may have positive societal impacts by making large pretrained models easier to reuse, combine, and analyze. If independently trained Transformers can be aligned and merged through low-loss linear paths, practitioners may be able to consolidate useful capabilities from multiple checkpoints without full retraining, reducing computational cost, energy consumption, and barriers to model development. The proposed symmetry-based analysis may also improve transparency by providing a more structured understanding of how different models relate in weight space, which could support safer model editing, auditing, and deployment.

The work may also have negative societal impacts. More effective model merging could make it easier to combine capabilities from different models, including potentially harmful capabilities, without extensive training resources. This may lower the barrier for creating models with stronger misuse potential or for obscuring the provenance of merged systems. In addition, if merged models are deployed without sufficient evaluation, they may inherit or amplify biases, unsafe behaviors, or failure modes from their source models. Moreover, if model merging becomes easier or more widely accessible, it may also increase the risk of intellectual property leakage. Proprietary capabilities, memorized content, or model-specific behaviors from a source checkpoint may be transferred to a merged model, while making attribution and ownership more difficult to verify. As a result, model owners may have reduced incentives to openly release their checkpoints due to concerns about uncontrolled reuse and merging.

## C Discussion: Why Does Dual Learned Matching Work?

A notable and perhaps surprising fact is that any dual-parameterized symmetry can be equivalently absorbed into a single-sided parameterization. We provide a formal proof of this equivalence in Appendix C.1. This result implies that dual parameterization does not introduce additional degrees of freedom, despite its empirical advantage over single-sided optimization. Therefore, the improved performance of dual learned matching cannot be attributed to increased parameter or symmetry capacity, but rather to the more favorable optimization geometry induced by overparameterization.

One hypothesis is that dual parameterization enables both checkpoints to move toward a shared, smoother region of the loss landscape, whereas single-sided optimization restricts movement to only one checkpoint, effectively anchoring the other. Moreover, dual parameterization allows complex symmetry transformations between two models to be decomposed into two simpler and more tractable transformations, thereby facilitating optimization.

### C.1 Proof of Dual-to-Single Absorption

We prove the equivalence by showing that the two endpoint transformations can be rewritten as one relative transformation plus a common functionality-preserving gauge. We first illustrate the reduction using the global residual-space symmetry. Let $\mathbf{G}_A, \mathbf{G}_B$ be the global orthogonal transformations applied to endpoints $A$ and $B$. Under the row-vector convention, a residual-stream representation transforms as $\mathbf{X} \mapsto \mathbf{X}\mathbf{G}$, so residual-reading and residual-writing matrices transform as

$$\mathbf{W}_{\text{in}} \mapsto \mathbf{G}^{\top}\mathbf{W}_{\text{in}}, \qquad \mathbf{W}_{\text{out}} \mapsto \mathbf{W}_{\text{out}}\mathbf{G}.$$

The dual interpolation gives

$$\widetilde{\mathbf{W}}_{\text{in}} = \lambda \mathbf{G}_A^\top \mathbf{W}_{\text{in}}^A + (1-\lambda)\mathbf{G}_B^\top \mathbf{W}_{\text{in}}^B, \qquad \widetilde{\mathbf{W}}_{\text{out}} = \lambda \mathbf{W}_{\text{out}}^A \mathbf{G}_A + (1-\lambda)\mathbf{W}_{\text{out}}^B \mathbf{G}_B.$$

Define the relative global transformation

$$\mathbf{R}_G := \mathbf{G}_B \mathbf{G}_A^\top.$$

Then $\mathbf{G}_B = \mathbf{R}_G \mathbf{G}_A$ and $\mathbf{G}_B^\top = \mathbf{G}_A^\top \mathbf{R}_G^\top$, so

$$\widetilde{\mathbf{W}}_{\text{in}} = \mathbf{G}_A^\top \underbrace{\left(\lambda \mathbf{W}_{\text{in}}^A + (1-\lambda)\mathbf{R}_G^\top \mathbf{W}_{\text{in}}^B\right)}_{\text{single-sided relative interpolation}},$$

$$\widetilde{\mathbf{W}}_{\text{out}} = \underbrace{\left(\lambda \mathbf{W}_{\text{out}}^A + (1-\lambda)\mathbf{W}_{\text{out}}^B \mathbf{R}_G\right)}_{\text{single-sided relative interpolation}} \mathbf{G}_A.$$

Because $\mathbf{G}_A$ and $\mathbf{G}_B$ are both orthogonal, their product $\mathbf{R}_G = \mathbf{G}_B \mathbf{G}_A^\top$ is also orthogonal. Thus the relative transformation $\mathbf{R}_G$ has exactly the same parameterized form as the original global residual-space symmetry parameter $\mathbf{G}$. Therefore the dual-interpolated layer is exactly the single-sided relative-interpolated layer under a valid global symmetry $\mathbf{R}_G$, followed by the same common global gauge $\mathbf{G}_A$ on the residual stream. Since this gauge is applied consistently to all residual reads and writes, it preserves the model function. Thus dual global matching is equivalent to single-sided relative global matching along the entire interpolation path.

For local circuit symmetries, the cancellation can be shown within each attention head by multiplying the paired matrices. For example, for the OV circuit of head $i$, define

$$\mathbf{R}_{V,i} := \mathbf{M}_{V,i}^B (\mathbf{M}_{V,i}^A)^{-1}.$$

Then the dual interpolation can be rewritten as

$$\widetilde{\mathbf{W}}_{V,i} = \big(\lambda \mathbf{W}_{V,i}^A + (1-\lambda)\mathbf{W}_{V,i}^B \mathbf{R}_{V,i}\big)\mathbf{M}_{V,i}^A, \qquad \widetilde{\mathbf{W}}_{O,i} = (\mathbf{M}_{V,i}^A)^{-1}\big(\lambda \mathbf{W}_{O,i}^A + (1-\lambda)\mathbf{R}_{V,i}^{-1}\mathbf{W}_{O,i}^B\big),$$

and hence

$$\widetilde{\mathbf{W}}_{V,i}\widetilde{\mathbf{W}}_{O,i} = \big(\lambda \mathbf{W}_{V,i}^A + (1-\lambda)\mathbf{W}_{V,i}^B \mathbf{R}_{V,i}\big) \underbrace{\mathbf{M}_{V,i}^A (\mathbf{M}_{V,i}^A)^{-1}}_{\mathbf{I}} \big(\lambda \mathbf{W}_{O,i}^A + (1-\lambda)\mathbf{R}_{V,i}^{-1}\mathbf{W}_{O,i}^B\big).$$

The same relative-parameter argument applies to the QK circuit in each head with

$$\mathbf{R}_{Q,i} := \mathbf{M}_{Q,i}^B (\mathbf{M}_{Q,i}^A)^{-1},$$

and to head and FFN permutations with

$$\mathbf{P}_{h,\ell} := \mathbf{P}_{h,\ell}^B (\mathbf{P}_{h,\ell}^A)^\top, \qquad \mathbf{P}_{f,\ell} := \mathbf{P}_{f,\ell}^B (\mathbf{P}_{f,\ell}^A)^\top.$$

Thus the dual-sided symmetry parameterization used in our method can be absorbed into a single-sided relative parameterization.

# D Experimental Details

We provide the details of the original pretrained models used in our study in Table 5. For each experiment, both models are from open-source checkpoints available on Hugging Face. For the ViT models, we adopt the `timm` version of the Google Research AugReg ViT checkpoints. The ImageNet-21k variants are pretrained on ImageNet-21k and fine-tuned on ImageNet-1k, whereas the ImageNet-1k variants are pretrained solely on ImageNet-1k. For the Pythia series, each model size is trained on the Pile [34] for approximately 300B tokens by EleutherAI; one checkpoint is trained on the original dataset, while the other is trained on a deduplicated version. For the HuggingFaceFW 1.8B ablation models, both checkpoints are trained for 350B tokens—one on the FineWeb-Edu [35] dataset and the other on the Pile.

We initialize the symmetry transformations with weight matching, solving the global transformation first, followed by the head and FFN permutations and the intra-head transformations in a non-iterative way. For symmetry learning, we optimize the continuous symmetry parameters using AdamW

Table 5: Endpoint checkpoints and architecture used in our experiments. Each endpoint pair in a row shares the listed architecture; $d_h$ is the attention head dimension.

| Model | Hugging Face Checkpoint | Arch. | $L$ | $d$ | $d_{\text{ff}}$ | $d_h$ | Norm. | Bias |
|---|---|---|---|---|---|---|---|---|
| ViT-S | A: `timm/vit_small_patch16_224.augreg_in21k_ft_in1k`<br>B: `timm/vit_small_patch16_224.augreg_in1k` | ViT | 12 | 384 | 1536 | 64 | LN | Yes |
| ViT-B | A: `timm/vit_base_patch16_224.augreg_in21k_ft_in1k`<br>B: `timm/vit_base_patch16_224.augreg_in1k` | ViT | 12 | 768 | 3072 | 64 | LN | Yes |
| ViT-L | A: `timm/vit_large_patch16_224.augreg_in21k_ft_in1k`<br>B: `glasses/vit_large_patch16_224` | ViT | 24 | 1024 | 4096 | 64 | LN | Yes |
| Pythia-14M | A: `EleutherAI/pythia-14m`<br>B: `EleutherAI/pythia-14m-deduped` | GPT-NeoX | 6 | 128 | 512 | 32 | LN | Yes |
| Pythia-70M | A: `EleutherAI/pythia-70m`<br>B: `EleutherAI/pythia-70m-deduped` | GPT-NeoX | 6 | 512 | 2048 | 64 | LN | Yes |
| Pythia-160M | A: `EleutherAI/pythia-160m`<br>B: `EleutherAI/pythia-160m-deduped` | GPT-NeoX | 12 | 768 | 3072 | 64 | LN | Yes |
| Pythia-410M | A: `EleutherAI/pythia-410m`<br>B: `EleutherAI/pythia-410m-deduped` | GPT-NeoX | 24 | 1024 | 4096 | 64 | LN | Yes |
| Pythia-1B | A: `EleutherAI/pythia-1b`<br>B: `EleutherAI/pythia-1b-deduped` | GPT-NeoX | 16 | 2048 | 8192 | 256 | LN | Yes |
| Pythia-1.4B | A: `EleutherAI/pythia-1.4b`<br>B: `EleutherAI/pythia-1.4b-deduped` | GPT-NeoX | 24 | 2048 | 8192 | 128 | LN | Yes |
| Pythia-6.9B | A: `EleutherAI/pythia-6.9b`<br>B: `EleutherAI/pythia-6.9b-deduped` | GPT-NeoX | 32 | 4096 | 16384 | 128 | LN | Yes |
| HF-1.8B | A: `HuggingFaceFW/ablation-model-fineweb-edu`<br>B: `HuggingFaceFW/ablation-model-the-pile` | LLaMA | 24 | 2048 | 8192 | 64 | RMSNorm | No |
| OLMo-7B | A: `allenai/OLMo-7B-hf`<br>B: `allenai/OLMo-7B-Twin-2T-hf` | OLMo | 32 | 4096 | 11008 | 128 | LN (non-param.) | No |

with a cosine learning rate scheduler. During training, we sample the interpolation coefficient as $\lambda \sim \mathcal{U}(a, b)$, with $a = 0.1$ and $b = 0.9$ for all models. For ViT models, training is conducted on ImageNet-1K with a batch size of 64. The learning rate starts at 1e-4 and gradually decreases to 2e-5 over 100,000 steps, equivalent to approximately 5 epochs. For language models, training is performed on WikiText-103 [26] with a batch size of 16 and a sequence length of 512. For the 14M-1.4B versions of Pythia, we use an initial learning rate of 5e-5, decreasing to 1e-5 over 50,000 steps. For Pythia-6.9B, the learning rate decreases from 3e-5 to 1e-5 over 20,000 steps, while for OLMo-7B, it decreases from 2e-5 to 5e-6 over 20,000 steps.

All models are trained using bf16 mixed precision, except for Pythia-14M and Pythia-70M, which use fp32 due to the instability of bf16. This instability may stem from unusually large logits, a known issue reported in the official repository. For perplexity evaluation, we use a sequence length of 512 without a sliding window and report the results on the WikiText-103 test set.

The entire symmetry learning procedure is lightweight. ViT and smaller Pythia models are trained on a single NVIDIA RTX 6000 Ada GPU for several hours to one day. For larger models, such as Pythia-6.9B and OLMo-7B, we utilize four NVIDIA RTX 6000 Ada GPUs, with training taking approximately one day.

The interpolation paths across model scales after LMC-DM are reported in Table 6. We also report the raw interpolation paths before matching in Table 7 as unaligned baselines.

# E Additional Experimental Results

## E.1 Less Related Endpoint Pairs

We evaluate Pythia-160M checkpoints trained with different random seeds, which are less related than the original/deduplicated pairs used in the main experiments (Table 8). Weight matching barely reduces the barrier, while LMC-DM achieves the lowest barrier.

Table 6: Interpolation paths across model scales after dual learned matching. ViT rows report ImageNet-1K accuracy (%), language-model rows report WikiText perplexity, and the rightmost column reports loss barrier.

| Model | Interpolation coefficient $\lambda$ | | | | | | | | | | | Loss Barrier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | |
| *ViT, Accuracy (%)* | | | | | | | | | | | | |
| Small | 81.40 | 81.08 | 79.01 | 74.52 | 66.42 | 57.10 | 56.13 | 64.51 | 72.71 | 77.71 | 78.68 | 1.09 |
| Base | 84.62 | 84.05 | 81.46 | 74.99 | 63.77 | 64.51 | 71.95 | 76.32 | 78.39 | 79.25 | 79.14 | 0.82 |
| Large | 85.82 | 85.89 | 85.30 | 83.96 | 80.62 | 73.96 | 69.40 | 75.67 | 80.43 | 82.67 | 82.97 | 0.66 |
| *Pythia, WikiText PPL* | | | | | | | | | | | | |
| 14M | 138.64 | 156.44 | 240.76 | 320.79 | 352.91 | 353.30 | 329.13 | 271.16 | 192.75 | 144.78 | 165.88 | 0.86 |
| 70M | 63.09 | 42.77 | 44.27 | 53.15 | 63.31 | 68.91 | 65.41 | 55.65 | 46.28 | 44.82 | 66.30 | 0.06 |
| 160M | 71.27 | 45.44 | 50.47 | 65.79 | 82.47 | 73.33 | 73.70 | 73.04 | 54.11 | 47.81 | 64.67 | 0.18 |
| 410M | 22.31 | 16.72 | 18.66 | 23.81 | 30.25 | 33.76 | 30.59 | 24.32 | 19.24 | 17.45 | 24.46 | 0.37 |
| 1B | 17.69 | 12.99 | 13.73 | 16.15 | 19.76 | 22.25 | 20.11 | 16.47 | 13.91 | 13.12 | 17.64 | 0.23 |
| 1.4B | 15.91 | 11.84 | 12.32 | 14.27 | 17.47 | 20.43 | 19.38 | 15.84 | 13.15 | 12.21 | 18.10 | 0.19 |
| 6.9B | 13.22 | 9.72 | 9.84 | 11.11 | 13.42 | 15.71 | 14.79 | 12.09 | 10.28 | 9.80 | 12.98 | 0.18 |
| *Other Models, WikiText PPL* | | | | | | | | | | | | |
| HF 1.8B | 15.71 | 11.73 | 12.26 | 14.90 | 20.02 | 26.72 | 24.02 | 17.16 | 13.15 | 11.86 | 15.01 | 0.55 |
| OLMo-7B | 10.83 | 9.84 | 12.60 | 20.68 | 42.19 | 51.80 | 25.16 | 15.41 | 11.32 | 10.09 | 11.61 | 1.53 |

Table 7: Raw interpolation paths between checkpoints before matching. Large perplexities are written in scientific notation.

| Model | Interpolation coefficient $\lambda$ | | | | | | | | | | | Loss Barrier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | |
| *ViT, Accuracy (%)* | | | | | | | | | | | | |
| Small | 81.46 | 75.88 | 32.43 | 0.89 | 0.16 | 0.13 | 0.12 | 0.27 | 6.08 | 62.60 | 78.71 | 6.58 |
| Base | 84.58 | 75.93 | 0.45 | 0.11 | 0.10 | 0.10 | 0.22 | 2.90 | 54.28 | 76.60 | 79.14 | 6.57 |
| Large | 85.82 | 84.25 | 65.73 | 0.46 | 0.09 | 0.10 | 0.10 | 0.06 | 6.36 | 69.28 | 83.00 | 6.47 |
| *Pythia, WikiText PPL* | | | | | | | | | | | | |
| 14M | 138.35 | 4.49e3 | 6.02e4 | 3.56e5 | 1.77e6 | 2.97e6 | 1.31e6 | 2.42e5 | 1.12e4 | 638.47 | 165.04 | 9.89 |
| 70M | 63.05 | 188.16 | 1.09e3 | 5.44e3 | 1.67e4 | 2.36e4 | 1.45e4 | 4.37e3 | 849.15 | 134.67 | 66.16 | 5.90 |
| 160M | 71.19 | 140.68 | 678.87 | 4.15e3 | 2.02e4 | 3.61e4 | 1.93e4 | 5.17e3 | 899.95 | 154.55 | 64.09 | 6.28 |
| 410M | 22.35 | 45.10 | 161.37 | 643.52 | 2.11e3 | 3.45e3 | 2.28e3 | 677.88 | 153.22 | 48.98 | 24.50 | 4.99 |
| 1B | 17.69 | 23.82 | 76.79 | 367.14 | 1.17e3 | 1.76e3 | 1.11e3 | 348.00 | 72.13 | 23.49 | 17.63 | 4.60 |
| 1.4B | 15.95 | 22.92 | 78.35 | 603.05 | 3.37e3 | 6.31e3 | 3.82e3 | 707.46 | 98.97 | 28.56 | 18.40 | 5.91 |
| 6.9B | 13.43 | 16.21 | 29.49 | 92.80 | 321.88 | 557.83 | 303.33 | 87.71 | 26.29 | 15.03 | 13.03 | 3.74 |
| *Other Models, WikiText PPL* | | | | | | | | | | | | |
| HF 1.8B | 15.71 | 19.22 | 116.63 | 8.62e3 | 2.07e4 | 2.79e4 | 2.40e4 | 1.26e4 | 177.02 | 18.55 | 15.01 | 7.50 |
| OLMo-7B | 10.80 | 13.63 | 153.74 | 5.38e3 | 3.64e4 | 3.06e4 | 5.66e3 | 734.08 | 34.20 | 12.95 | 11.45 | 8.10 |

Table 8: Loss barriers on less related endpoint pairs. Pythia-160M: `pythia-160m` vs. `pythia-160m-seed1` (different random seeds).

| Model pair | Raw | WM | Learned Matching | LMC-DM |
|---|---|---|---|---|
| Pythia-160M (default / seed1) | 11.41 | 10.00 | 1.06 | **0.85** |

### E.2 Generalization Beyond WikiText

Table 9 evaluates the midpoint models on C4 [36] and the Pile [34], using symmetry transformations learned only on WikiText-103. LMC-DM achieves the lowest midpoint perplexity in all cases, although a gap to the endpoints remains.

Table 9: Midpoint ($\lambda = 0.5$) perplexity on C4 and the Pile. Symmetry transformations are learned on WikiText-103 and kept frozen; no additional training is performed on C4 or the Pile. Endpoint rows report the two original checkpoints (A / B).

| Model pair | Method | C4 PPL ↓ | Pile PPL ↓ |
|---|---|---|---|
| Pythia-410M (original / deduped) | Endpoints (A / B) | 23.32 / 24.78 | 11.44 / 12.29 |
| | Raw | 1990.54 | 1345.85 |
| | Learned Matching | 298.77 | 299.61 |
| | LMC-DM | **270.22** | **291.30** |
| Pythia-1B (original / deduped) | Endpoints (A / B) | 19.74 / 19.64 | 9.83 / 9.93 |
| | Raw | 1361.34 | 1013.40 |
| | Learned Matching | 222.82 | 270.12 |
| | LMC-DM | **170.86** | **206.86** |
| HF-1.8B (FineWeb-Edu / Pile) | Endpoints (A / B) | 18.26 / 17.29 | 15.71 / 7.48 |
| | Raw | 22499.11 | 26625.51 |
| | LMC-DM | **211.59** | **269.97** |

### E.3 Downstream Zero-Shot Tasks

Table 10 gives a preliminary downstream check on zero-shot commonsense tasks: PIQA [37], SciQ [38], and ARC-Easy [39]. Although LMC-DM improves over raw interpolation, the matched interpolated models still lag behind the original endpoints, indicating that low WikiText barriers do not yet imply lossless downstream behavior. This is a limitation of the current setup: Pythia is a base model, and due to limited resources, our symmetry learning uses only hundreds of millions of tokens without downstream supervision; scaling this training budget may improve downstream preservation, which we leave to future work.

Table 10: Zero-shot commonsense accuracy for Pythia endpoints and matched models.

| Model | Method | PIQA | SciQ | ARC-E |
|---|---|---|---|---|
| 410M | Original Endpoint | 66.43 | 81.50 | 51.89 |
| 410M | Raw Interpolation | 55.39 | 31.50 | 29.71 |
| 410M | Learned Matching | 55.44 | 56.30 | 33.25 |
| 410M | Dual Learned Matching | 55.71 | 60.00 | 35.69 |
| 1B | Original Endpoint | 70.95 | 83.90 | 56.73 |
| 1B | Raw Interpolation | 54.35 | 35.50 | 29.08 |
| 1B | Learned Matching | 55.98 | 64.80 | 36.57 |
| 1B | Dual Learned Matching | 56.42 | 64.40 | 37.16 |

### E.4 Seed Variability

Table 11 reports LMC-DM over independent symmetry-optimization seeds. The standard deviation is small for both models, indicating that the results are insensitive to the optimization seed.

Table 11: Loss barrier (mean $\pm$ standard deviation) of LMC-DM over independent symmetry-optimization seeds. Endpoints, weight-matching initialization, and evaluation set are fixed.

| Model | Barrier ↓ |
|---|---|
| Pythia-14M | $0.8637 \pm 0.0015$ |
| Pythia-70M | $0.0631 \pm 0.0011$ |

### E.5 Computational Overhead

Table 12 reports the per-step wall-clock overhead of optimizing both endpoints instead of one. The overhead is modest, ranging from 3.4% to 11.5%.

Table 12: Per-step wall-clock overhead of LMC-DM relative to Enhanced Learned Matching.

| | ViT-B | Pythia-410M | Pythia-1B |
|---|---|---|---|
| Overhead | 11.5% | 3.4% | 8.2% |

# F Normalization Reparameterization Details

This appendix gives the algebra behind the normalization reparameterization used in Section 2. We follow the same row-vector convention as in the main text. Let

$$\mathbf{C} = \mathbf{I} - \frac{1}{d}\mathbf{1}\mathbf{1}^\top$$

be the channel-centering operator. Writing $\mathrm{LayerNorm}_0$ and $\mathrm{RMSNorm}_0$ for the corresponding parameter-free normalization operators, the non-affine part of LayerNorm can be written as RMSNorm on the centered residual stream:

$$\mathrm{LayerNorm}_0(\mathbf{X}) = \mathrm{RMSNorm}_0(\mathbf{X}\mathbf{C}).$$

Thus a pre-LayerNorm Transformer can be represented in a parameter-free pre-RMSNorm form by keeping residual writes centered. If a branch writes to the residual stream as

$$\mathbf{Y} = \mathbf{U}\mathbf{W}_{\mathrm{out}} + \mathbf{1}\mathbf{b}_{\mathrm{out}}^\top,$$

then its centered form is

$$\mathbf{Y}\mathbf{C} = \mathbf{U}(\mathbf{W}_{\mathrm{out}}\mathbf{C}) + \mathbf{1}(\mathbf{C}\mathbf{b}_{\mathrm{out}})^\top.$$

Therefore the final weight matrix of each residual-writing branch is projected as $\mathbf{W}_{\mathrm{out}}\mathbf{C}$. If a bias is present, only its all-ones component is removed, while the centered component is kept or absorbed into an adjacent affine term.

The affine parameters of LayerNorm or RMSNorm can also be absorbed into the next linear layer. Let $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ be the normalization gain and bias, and let the following affine map have weight $\mathbf{W}$ and bias $\mathbf{b}$. Then

$$\begin{aligned}\mathrm{LayerNorm}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{X})\mathbf{W} + \mathbf{1}\mathbf{b}^\top &= \mathrm{RMSNorm}_0(\mathbf{X}\mathbf{C})\mathrm{Diag}(\boldsymbol{\gamma})\mathbf{W} + \mathbf{1}(\mathbf{b} + \mathbf{W}^\top\boldsymbol{\beta})^\top \\ &= \mathrm{RMSNorm}_0(\mathbf{X}\mathbf{C})\mathbf{W}' + \mathbf{1}\mathbf{b}'^\top,\end{aligned}$$

where

$$\mathbf{W}' = \mathrm{Diag}(\boldsymbol{\gamma})\mathbf{W}, \qquad \mathbf{b}' = \mathbf{b} + \mathbf{W}^\top\boldsymbol{\beta}.$$

# G Continuous Symmetry Parameterizations

This section gives additional details on the continuous symmetry parameterizations used in LMC-DM. In all cases, the goal is not merely to introduce learnable variables, but to keep the transformed weights inside the valid functionality-preserving symmetry family throughout gradient-based optimization. In our implementation, both endpoint models maintain their own symmetry modules, initialized from weight matching when available.

## G.1 Cayley Transform with Signs

For orthogonal symmetries, we mainly use the Cayley transform with a fixed sign component. Given a free matrix $\mathbf{S}$, we form a skew-symmetric matrix

$$\mathbf{A} = \mathbf{S} - \mathbf{S}^\top,$$

and parameterize the smooth orthogonal factor as

$$\mathbf{C}(\mathbf{A}) = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{I} + \mathbf{A}).$$

This guarantees $\mathbf{C}(\mathbf{A})^\top\mathbf{C}(\mathbf{A}) = \mathbf{I}$ at every optimization step, avoiding the need for projection after each update. In the implementation, the signed variant composes this standard Cayley factor with a fixed signed left factor chosen at initialization, which allows us to represent weight matching solutions with positive or negative determinant without changing the optimization formula.

### G.2 Matrix Exponential Parameterization

We use two matrix-exponential parameterizations, depending on whether the symmetry is constrained to be orthogonal or merely invertible. For orthogonal symmetries, the exponent is skew-symmetric:

$$\mathbf{A} = \mathbf{S} - \mathbf{S}^\top, \qquad \mathbf{Q} = \exp(\mathbf{A}), \qquad \mathbf{Q}^{-1} = \mathbf{Q}^\top,$$

with an optional fixed reflector handled in the implementation to match the determinant component selected at initialization. For general invertible symmetries, the exponent is unconstrained:

$$\mathbf{M} = \exp(\mathbf{U}), \qquad \mathbf{M}^{-1} = \exp(-\mathbf{U}),$$

where $\mathbf{U}$ is a free matrix. The implementation again optionally composes this standard form with a fixed reflector when needed. This keeps $\mathbf{M}$ invertible by construction and gives an exact inverse.

### G.3 Polar Parameterization

For invertible symmetries, the polar parameterization provides a promising alternative to the matrix exponential parameterization. We decompose the symmetry matrix into an orthogonal factor and a positive-definite factor,

$$\mathbf{M} = \mathbf{Q}\mathbf{P},$$

where $\mathbf{Q}$ is an orthogonal factor parameterized by the Cayley transform previously discussed, and $\mathbf{P}$ is positive definite. The positive-definite factor $\mathbf{P}$ is represented as

$$\mathbf{P} = \mathbf{L}\mathbf{L}^\top,$$

where $\mathbf{L}$ is lower triangular with positive diagonal entries. This parameterization keeps the matrix invertible, provides a stable inverse

$$\mathbf{M}^{-1} = \mathbf{P}^{-1}\mathbf{Q}^\top,$$

and separates rotational and scaling/shearing degrees of freedom, which we found more stable than unconstrained direct matrices in some settings.

### G.4 Potential Alternative Parameterizations

We also considered several other alternatives. Direct invertible parameterization optimizes a free matrix and computes its inverse explicitly; it is simple and expressive, but can become ill-conditioned during training. For orthogonal symmetry, SVD projection parameterizes an unconstrained matrix and projects it to the nearest orthogonal factor in the forward pass; while flexible, this introduces an expensive projection and can lead to instability in backpropagation. Householder products provide a structured orthogonal parameterization, but we found them relatively slow on large matrices in GPU-based training.

## H Further Discussion

**Insights from Linear Mode Connectivity.** Our study demonstrates that independently trained checkpoints exhibit linear mode connectivity when accounting for weight-space symmetries. This suggests that neural networks converge to functionally similar solutions within a structured, low-loss region of the parameter space, rather than isolated minima. These findings align with the Platonic Representation Hypothesis [40], suggesting that representational convergence extends into the weight space itself. This perspective reveals a significant degree of structural redundancy that is often underestimated when models are viewed only through their raw parameterization.

**Potential Applications.** We provide several potential applications of the proposed LMC-DM method:

- **Weight Space Learning and Generation:** By resolving symmetries between different checkpoints, LMC-DM produces a family of aligned models that share a consistent weight-space structure. This may facilitate training generative models over neural network weights. In addition, low-loss interpolation paths themselves provide a principled way to sample new models, which could be useful for model ensembling or as training data for weight-space generative methods.

- **Federated Learning:** Symmetry-based alignment can reduce inter-client variance by mapping locally trained models into a shared coordinate system, potentially improving aggregation quality and convergence stability in federated learning.
- **Efficient AI:** If independently trained models can be reliably merged without performance degradation, it becomes possible to consolidate capabilities from multiple checkpoints without full retraining. This could significantly reduce computational cost, energy consumption, and barriers to large-scale model development.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope of the work: symmetry-based dual learned matching for linear mode connectivity in pretrained Transformers, with claims supported by the method description in Section 2–3.3 and the empirical results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the symmetry in Section 2 and provides the algebraic details in Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 describes the evaluation protocol and metrics, and Appendix D lists the model checkpoints, datasets, hyperparameters, etc.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The experiments use publicly available model checkpoints and datasets, and the paper provides detailed experimental settings in Appendix D. We will release the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies the datasets, metrics, interpolation protocol, baselines, and ablations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the standard deviation over independent seeds in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D reports the GPU type and approximate runtime.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more computing than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available checkpoints and datasets, does not involve human subjects, and discusses limitations and societal impacts in Appendix. We are not aware of any aspect of the research that deviates from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have included them in Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce or release new pretrained language models, image generators, or scraped datasets. The experiments operate on existing publicly available checkpoints and datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in this work are properly credited, and their licenses and terms of use are respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset, benchmark, or pretrained model checkpoint as a new asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are only used for refining the writing of the paper, and do not impact the core methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.