STRUCTURED MODEL EMBEDDINGS AS REPRESENTATIONS OF TASK ADAPTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning a shared pretrained model yields a collection of task-specific parameter updates that, beyond serving as deployable adaptations, can provide a representation of learned task behavior. A useful representation should preserve adaptation capacity, capture meaningful task relationships, and support interpretable functional changes. We investigate these properties through a structured *model embedding space* whose coordinates are induced by the pretrained model. Each model embedding specifies a parameter update through fixed bases, providing both a compact representation of an adaptation and a direct mechanism for operating on it. We study four key properties of this space: *adaptation fidelity, relational geometry, functional component structure,* and *operational usefulness*. We theoretically characterize the resulting coordinate system and connect embedding-space changes to functional changes and task arithmetic. We then examine these properties in a controlled MNIST generation setting with explicitly defined task relations, followed by larger-scale evaluations using ViT across 20 image datasets.

1 INTRODUCTION

Many downstream tasks are learned from the same pretrained model, producing a growing collection of task-specific model variants. Parameter-efficient fine-tuning makes it practical to store these variants as lightweight adapters (Hu et al., 2022; Houlsby et al., 2019a; Li & Liang, 2021; Liu et al., 2024). These adaptations record how a common pretrained model changes in response to different tasks. This raises a more fundamental question: *can these learned parameter changes themselves provide a representation of tasks?*

This question connects model adaptation to the broader goal of representation learning, which seeks representations that expose useful structure for subsequent comparison and manipulation (Bengio et al., 2013; Mikolov et al., 2013). In this case, however, the objects being represented are not individual data samples, but learned tasks. A parameter-based task representation has a distinctive advantage: it remains directly connected to model behavior. A representation can therefore be used not only to compare tasks, but also to modify models and observe the resulting behavioral changes.

The main challenge is that the raw parameter space is not naturally a meaningful task space. Fine-tuned models contain millions of interacting parameters, many of which may encode redundant or weakly relevant variation (Hu et al., 2022; Ilharco et al., 2023). Moreover, parameter changes obtained for different tasks need not admit an immediately interpretable geometry: proximity between two updates does not necessarily imply similar behavior, and algebraic operations between them do not necessarily correspond to meaningful task-level changes. Thus, simply treating arbitrary parameter updates as task representations is insufficient.

Hypothesis. Based on this perspective, we study the following hypothesis: Tasks can be represented in a shared, structured subspace of a pretrained model's parameter space. The geometry of this subspace reveals meaningful relationships among tasks by reflecting how different directions change the model's behavior on data.

To evaluate our hypothesis, we construct a structured *model embedding space*: a shared parameter subspace in which each task adaptation is represented by a compact embedding, as illustrated in Figure 1. We then assess whether this space serves as a meaningful representation of learned task adaptations through three complementary forms of evidence.

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

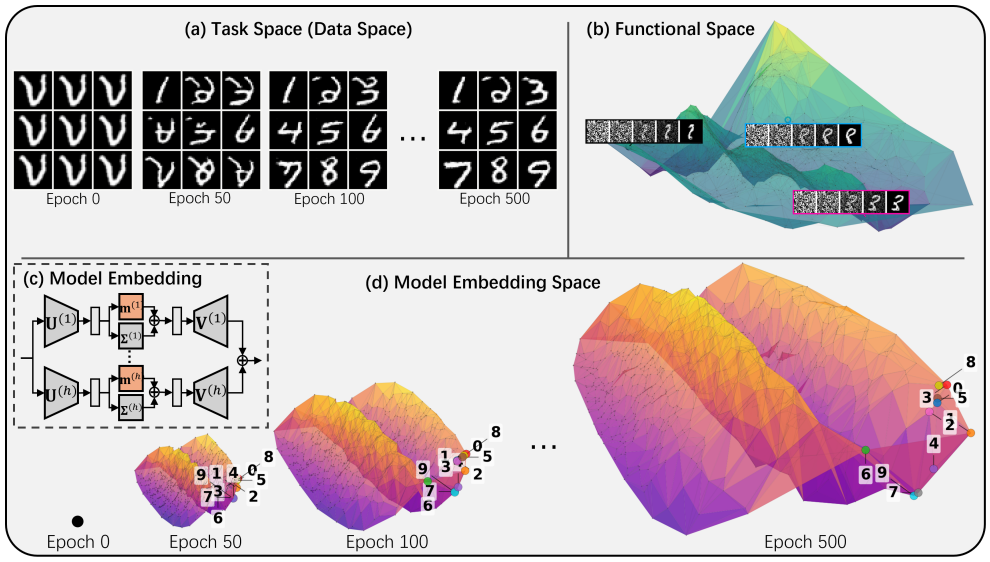


Figure 1: **Model Embeddings (MODEMs) as task representations.** Different from the representation in (a) data space, we study the properties of MODEMs via (b) functional relations. (c) The *MODEM* $\mathbf{m}$ represents the update encoded relative to the pretrained model. With respect to the fixed pretrained bases $\mathbf{U}$ and $\mathbf{V}$, MODEMs serve as coordinates that parameterize these updates within the pretrained model subspace. (d) We visualize the *MODEM space* by treating each $\mathbf{m}$ as a point in the shared coordinate space. **Interpretation:** Adaptation in the (d) MODEM space encodes task-specific knowledge, inducing corresponding changes in the generated (a) data space. Our study focuses on whether and how this space provides a meaningful representation of learned task adaptations. Details are in Section 2.

1. Section 2.2 provides a functional interpretation of model embeddings by relating changes in the embedding space to changes in the induced model function.
2. Section 3.2 demonstrates that the geometry of model embeddings aligns with independently defined task relationships across controlled and large-scale settings, providing evidence that the embedding space captures meaningful task structure.
3. Section 3.3 demonstrates that this structure is functionally actionable: individual embedding components induce distinguishable behavioral effects, while addition and subtraction in the embedding space produce corresponding changes in task adaptations.

2 Model Embeddings as Task Representations

Let $f_{\mathbf{W}_0}$ denote a model with fixed pretrained parameters $\mathbf{W}_0$. A task t is specified by a data distribution P_t and an objective

$$\mathcal{L}_t(\Delta\mathbf{W}_t) = \mathbb{E}_{z\sim P_t}\left[\ell_t\left(f_{\mathbf{W}_0+\Delta\mathbf{W}_t}; z\right)\right], \tag{1}$$

where $z \sim P_t$ denotes a task sample and ℓ_t is the corresponding per-sample loss. Starting from the common pretrained model $\mathbf{W}_0$, adaptation to task t produces a parameter update $\Delta\mathbf{W}_t$. The resulting task-specific model is therefore $f_{\mathbf{W}_0+\Delta\mathbf{W}_t}$. The summary of notation is in Appendix Table 5.

2.1 Model Embeddings as Task Representations

Rather than representing a task by the full parameter update $\Delta\mathbf{W}_t$, we introduce a low-dimensional model embedding (MODEM) $\mathbf{m}_t$ and a fixed decoder D such that

$$\Delta\mathbf{W}_t = D(\mathbf{m}_t). \tag{2}$$

$\mathbf{m}_t$ provides coordinates for the learned adaptation relative to the common pretrained model $\mathbf{W}_0$.

Constructing the decoder D. Consider an adaptable pretrained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We construct a fixed decoder from the structure of $\mathbf{W}_0$. Let

$$\mathbf{W}_0 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{3}$$

be its singular value decomposition, with $r = \min(d_{\text{out}}, d_{\text{in}})$ denoting the total number of available singular directions. We choose a block dimension r_h and partition the r singular directions into $H = r/r_h$ disjoint blocks, assuming divisibility for simplicity. The h-th block contains

$$\mathbf{U}^{(h)} \in \mathbb{R}^{d_{\text{out}} \times r_h}, \qquad \mathbf{V}^{(h)} \in \mathbb{R}^{d_{\text{in}} \times r_h}. \tag{4}$$

The H blocks define the full set of available pretrained directions from which an adaptation can be constructed. For task t, each selected block h is associated with a trainable coordinate matrix

$$\mathbf{M}_t^{(h)} \in \mathbb{R}^{r_h \times r_h}. \tag{5}$$

Let $\mathcal{H}_t \subseteq \{1, \ldots, H\}$ denote the set of blocks used for the adaptation. The decoder maps the MODEM to the parameter update as

$$\Delta\mathbf{W}_t = D(\mathbf{m}_t) = \sum_{h \in \mathcal{H}_t} \mathbf{U}^{(h)} \mathbf{M}_t^{(h)} \mathbf{V}^{(h)\top}. \tag{6}$$

Model embedding (MODEM). The complete MODEM for task t is obtained by a collection of trainable coordinate matrices across all layers $\mathcal{J} \subseteq \{1, \ldots, J\}$ and blocks $\mathcal{H} \subseteq \{1, \ldots, H\}$,

$$\mathbf{m}_t = \left(\left\{ \mathbf{M}_{t,j}^{(h)} : j \in \mathcal{J},\ h \in \mathcal{H} \right\} \right). \tag{7}$$

Definition 1 (MODEM space)**.** *For fixed pretrained parameters $\mathbf{W}_0$, fixed decoders D, and a fixed set of permitted coordinates, define the coordinate space $\mathcal{C} = \mathbb{R}^{J \times H \times r_h \times r_h}$. The corresponding* MODEM update subspace *is*

$$\mathcal{E}_{\mathbf{W}_0} = \{ D(\mathbf{m}) : \mathbf{m} \in \mathcal{C} \}, \tag{8}$$

and the associated family of adapted models is $\mathbf{W}(\mathbf{m}) = \mathbf{W}_0 + D(\mathbf{m})$. The zero embedding $\mathbf{m} = \mathbf{0}$, produces no update and therefore represents the pretrained model $\mathbf{W}_0$.

When a vector representation is required, we write $\text{vec}(\mathbf{m}) \in \mathbb{R}^p$, where $p = JHr_h^2$ is the number of parameters in the MODEM. Because D is linear in the coordinates, $\mathcal{E}_{\mathbf{W}_0}$ is a linear subspace of the ambient parameter-update space. The decoder is fixed after pretraining and reused across tasks.

Represent a task through MODEMs. Given a training set $S_t = \{z_i\}_{i=1}^{n_t}, z_i \sim P_t$ of task t, we obtain an embedding by adapting within the MODEM space:

$$\mathbf{m}_t = \mathcal{A}(S_t, \mathbf{W}_0, D), \qquad \widehat{\mathcal{L}}_t(\mathbf{m}) = \frac{1}{|S_t|} \sum_{z_i \in S_t} \ell_t(f_{\mathbf{W}(\mathbf{m})}; z_i), \tag{9}$$

where $\mathcal{A}$ denotes a training protocol, $\widehat{\mathcal{L}}_t$ is empirical objective, and ℓ_t is the per-sample loss.

2.2 Properties of MODEM Representations

Structure of the MODEM space. MODEMs form a linear, low-dimensional coordinate space in which the decoder preserves the dimensionality and pairwise distances of the represented updates.

Proposition 1 (Linearity, dimension, and distance preservation)**.** *For disjoint orthonormal blocks in (6), D is an injective linear map, and*

$$\begin{aligned} D(a\mathbf{m}_i + b\mathbf{m}_j) &= aD(\mathbf{m}_i) + bD(\mathbf{m}_j), \\ \dim(\mathcal{E}_{\mathbf{W}_0}) &= p = JHr_h^2, \\ \|D(\mathbf{m}_i) - D(\mathbf{m}_j)\|_F &= \|\mathbf{m}_i - \mathbf{m}_j\|_F. \end{aligned}$$

These properties establish faithful coordinates for the permitted updates.

Relational geometry and functional behavior. Recall the model $f(x;\mathbf{m}) = f_{\mathbf{W}(\mathbf{m})}(x)$ and its functional Jacobian $\Phi(x;\mathbf{m}) := \frac{\partial f(x;\mathbf{m})}{\partial \mathbf{m}}$. Consider the straight adaptation path $\mathbf{m}_t(\alpha) = \alpha \mathbf{m}_t$, with $\alpha \in [0,1]$, partitioned as $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_K = 1$.

Proposition 2. *Let $\mathbf{m}_t^{(k)} := \alpha_k \mathbf{m}_t$ and $\Delta\mathbf{m}_t^{(k)} := \mathbf{m}_t^{(k)} - \mathbf{m}_t^{(k-1)}$. The task-specific functional change can be approximated by*

$$f(x;\mathbf{m}_t) - f_0(x) \approx \sum_{k=1}^{K} \Phi\Big(x;\mathbf{m}_t^{(k-1)}\Big)\,\Delta\mathbf{m}_t^{(k)},$$

where $f_0(x) = f_{\mathbf{W}_0}(x)$ denotes the pretrained model.

The functional change associated with task t can be represented as the accumulation of local linear changes along the path from the pretrained model $\mathbf{0}$ to its MODEM $\mathbf{m}_t$.

Define the path-averaged functional map $\bar{\Phi}_t(x) := \sum_{k=1}^{K} (\alpha_k - \alpha_{k-1}) \Phi\Big(x;\mathbf{m}_t^{(k-1)}\Big)$, then

$$f(x;\mathbf{m}_t) - f_0(x) \approx \bar{\Phi}_t(x)\mathbf{m}_t. \tag{10}$$

Theorem 1 (Task relations)**.** *For two tasks t_i and t_j,*

$$f(x;\mathbf{m}_i) - f(x;\mathbf{m}_j) \approx \bar{\Phi}_i(x)\ \underbrace{(\mathbf{m}_i - \mathbf{m}_j)}_{\textit{MODEM difference}} + \underbrace{\big[\bar{\Phi}_i(x) - \bar{\Phi}_j(x)\big]}_{\textit{functional geometry mismatch}}\ \mathbf{m}_j.$$

Corollary 2 (Shared local geometry)**.** *If the two tasks traverse similar functional geometry, $\bar{\Phi}_i(x) \approx \bar{\Phi}_j(x) \approx \bar{\Phi}(x)$, then*

$$f(x;\mathbf{m}_i) - f(x;\mathbf{m}_j) \approx \bar{\Phi}(x)(\mathbf{m}_i - \mathbf{m}_j).$$

If the adaptation paths remain close to the pretrained functional geometry, $\bar{\Phi}(x) \approx \Phi_0(x)$, this further reduces to

$$f(x;\mathbf{m}_i) - f(x;\mathbf{m}_j) \approx \Phi_0(x)(\mathbf{m}_i - \mathbf{m}_j).$$

Hence, under shared local geometry, differences between MODEMs directly encode differences between task functions.

Remark 3. *Theorem 1 and Corollary 2 establish a direct connection between MODEM space and task space: changes in embedding coordinates induce corresponding changes in model behavior.*

Detailed assumptions, analysis, and proofs are provided in Appendix D.

2.3 Operations in MODEM Space

Because task representations are expressed in a shared coordinate system, we can manipulate them directly and study how these operations propagate to the model function. Given task embeddings $\{\mathbf{m}_i\}_{i=1}^{N}$, consider a general linear operation

$$\mathbf{m}_{\text{op}} := \sum_{i=1}^{N} \lambda_i \mathbf{m}_i, \tag{11}$$

where $\lambda_i \in \mathbb{R}$ are operation coefficients. From Corollary 2,

$$f(x;\mathbf{m}_{\text{op}}) - f_0(x) \approx \bar{\Phi}(x)\mathbf{m}_{\text{op}} = \sum_{i=1}^{N} \lambda_i \bar{\Phi}(x)\mathbf{m}_i \approx \sum_{i=1}^{N} \lambda_i \big(f(x;\mathbf{m}_t) - f_0(x)\big). \tag{12}$$

Thus, under approximately shared functional geometry, linear operations in MODEM space induce corresponding linear combinations of task-specific functional changes.

Addition. For two task embeddings $\mathbf{m}_a$ and $\mathbf{m}_b$, addition is defined as

$$\mathbf{m}_{a+b} := \mathbf{m}_a + \mathbf{m}_b. \tag{13}$$

When the relevant embeddings share similar functional geometry,

$$f(x;\mathbf{m}_{a+b}) - f_0(x) \approx \big(f(x;\mathbf{m}_a) - f_0(x)\big) + \big(f(x;\mathbf{m}_b) - f_0(x)\big). \tag{14}$$

Hence, addition in embedding space corresponds approximately to combining the functional changes associated with the two tasks. More generally, interpolation or weighted merging can be written as $\mathbf{m}_{\text{merge}} = \lambda\mathbf{m}_a + (1-\lambda)\mathbf{m}_b, \lambda \in [0,1]$.

Table 1: **MODEMs serve as effective representations of task adaptations**: With a small number of trainable parameters, MODEMs capture the task-specific knowledge required to generate the target digits. We report the digit classification accuracy of the generated images.

Method	Params.	Target digits ↑	Other digits ↓
Zero-shot	0	1.51%	1.62%
Full tuning	789,504	**91.57%**	**5.04%**
LoRA	49,152	88.38%	6.86%
MODEM	**24,576**	<u>88.63%</u>	<u>5.41%</u>

Subtraction. Subtraction is defined as

$$\mathbf{m}_{a-b} := \mathbf{m}_a - \mathbf{m}_b. \tag{15}$$

Under similar functional geometry,

$$f(x; \mathbf{m}_{a-b}) - f_0(x) \approx \big(f(x; \mathbf{m}_a) - f_0(x)\big) - \big(f(x; \mathbf{m}_b) - f_0(x)\big), \tag{16}$$

so subtraction in embedding space approximately removes the functional change associated with $\mathbf{m}_b$ from that associated with $\mathbf{m}_a$. More general displacement transfer can be written as $\mathbf{m}_{\text{op}} = \mathbf{m}_c + \lambda(\mathbf{m}_b - \mathbf{m}_a)$, where the displacement $\mathbf{m}_b - \mathbf{m}_a$ is applied to $\mathbf{m}_c$. When the corresponding regions share similar functional geometry, this displacement induces an analogous change in function space.

3 Controlled Study: MNIST Task Representations

We first study MODEMs in a controlled setting where task relations are explicitly known. Let $S \subseteq \{0, \ldots, 9\}$ denote a nonempty subset of MNIST digits. We define task t_S by the uniform mixture $P_{t_S}(x) = \frac{1}{|S|} \sum_{i \in S} P_i(x)$, where P_i is the distribution of digit i. The complete study contains all $2^{10} - 1 = 1023$ nonempty subsets. This provides a controlled task family in which task relations can be measured directly from overlap between digit sets. We pretrain a transformer on EMNIST letters (no digits) and independently adapt it to each MNIST task using MODEMs $\mathbf{m}_t$. Experimental details are provided in Appendix B.

3.1 Task Adaptation Capability

We evaluated whether adaptation changes an EMNIST-letter generator into a generator of the requested MNIST digits. All adaptation methods used 500 MNIST training examples per target digit and a frozen output head. MODEM updated the attention value projections; rank-8 LoRA updated the attention projections; full attention tuning updated their ordinary weights.

MODEMs substantially acquire the new digit distribution. An independent 47-class classifier, trained to distinguish 10 digits and 37 EMNIST letter classes, labels the generated images. Its held-out accuracy was 86.98% on EMNIST Balanced and 99.09% on MNIST. We report the fraction classified as target digits and other digits. Table 1 reports mean accuracy across all tasks. MODEM increased the mean target accuracy to 88.63% while training 24,576 parameters.

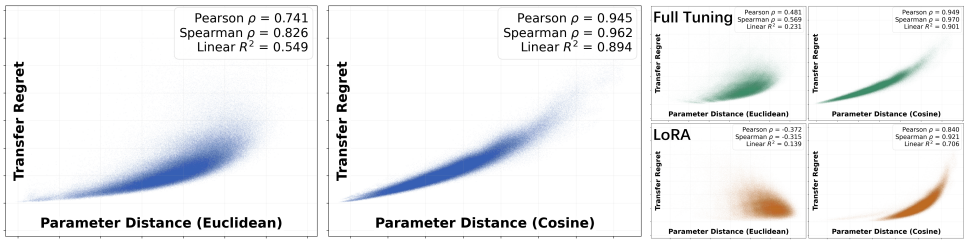
3.2 Task Relationships and Visualization

For each adaptation method, we compared distances between pairs of task-specific parameter updates with task relations, measured by zero-shot transfer regret (18). We provide additional task relation results using digit-set overlap and image-distribution discrepancy in the Appendix B. Correlations were computed over $1{,}023(1{,}022)/2 = 522{,}753$ task pairs.

Parameter distances. Recall in Corollary 2, we measure the Euclidean distance and cosine distance between two MODEMs via

$$d_{\text{Euc}}(i,j) = \| \operatorname{vec}(\mathbf{m}_i) - \operatorname{vec}(\mathbf{m}_j) \|_2, \quad d_{\cos}(i,j) = 1 - \frac{\operatorname{vec}(\mathbf{m}_i)^\top \operatorname{vec}(\mathbf{m}_j)}{\| \operatorname{vec}(\mathbf{m}_i) \|_2 \| \operatorname{vec}(\mathbf{m}_j) \|_2}. \tag{17}$$

Distances are computed within each adaptation method; the raw parameter vectors of different methods have different coordinate systems.



(a) (b) (c)

Figure 2: **MODEM geometry is strongly correlated with task relationships in both angular and Euclidean structure.** We quantify *task relation* using zero-shot transfer regret, which measures how effectively an embedding learned for one task transfers to another without further adaptation. Distances in the MODEM space are characterized using (a) Euclidean distance and (b) cosine distance, capturing norm-sensitive and angular geometric relationships, respectively. (c) shows the correlation between task-space relationships and parameter-space geometry under full fine-tuning and LoRA. Comparisons with other task relation metrics are in Table 6.

Table 2: **Operations (addition and subtraction) in model-embedding space have an effective impact on task space.** Each entry reports $(p_a/p_b/p_c)$, where p_a, p_b, and p_c denote the frequencies of generating digits a, b, and c, respectively. For example, a "Pair" adapter may mainly generate digits "0" and "1", while adding a "Singleton" increases the generation probability of digit "2", effectively combining the corresponding knowledge into the constructed adapter.

Addition	**Pair** (*e.g.* digit 0 1)	**Singleton** (*e.g.* digit 2)	**Constructed** (*e.g.* digit 0 1 2)
Distribution	43.2%/45.0%/1.5%	0.7%/0.9%/92.7%	**29.4%** ↓ /**32.0%** ↓ /**15.4%** ↑
Subtraction	**Triplet** (*e.g.* digit 0 1 2)	**Pair** (*e.g.* digit 0 1)	**Constructed** (*e.g.* digit 2)
Distribution	28.9%/29.0%/28.9%	43.2%/45.0%/1.5%	**2.4%** ↓ /**1.6%** ↓ /**82.0%** ↑

Task relation. We evaluate the task relation using zero-shot transfer regret. Recall the empirical objective (9), we define the loss of $\mathbf{m}_i$ on target task j as $\widehat{\mathcal{L}}_j(\mathbf{m}_i)$. Using the corresponding pretrained-model loss $\widehat{\mathcal{L}}_j(\mathbf{0})$, we define directed normalized regret as

$$R_{i\to j} = \frac{\widehat{\mathcal{L}}_j(\mathbf{m}_i) - \widehat{\mathcal{L}}_i(\mathbf{m}_j)}{|\widehat{\mathcal{L}}_j(\mathbf{0}) - \widehat{\mathcal{L}}_j(\mathbf{m}_j)|}. \tag{18}$$

MODEMs as task representations. Figure 1 visualizes the task representations in the MODEM space: It illustrates both the relational structure of task representations and how their functional effects evolve during optimization. We visualize this space by projecting adapters from all tasks and fine-tuning epochs into a single PCA basis estimated jointly from their MODEMs. Each point represents the adapter of a particular task at a particular training epoch, so point locations are directly comparable, and their movement reflects changes in the learned representations.

MODEMs have strong correlation with task relations. Figure 2 shows a near-linear correlation between transfer regret and MODEM distance. Cosine distance between MODEMs is highly correlated with transfer regret, with both Pearson and Spearman correlations reaching about 0.95. Euclidean distance also exhibits substantial correlation, with Pearson and Spearman coefficients of 0.74 and 0.83, respectively. These results indicate that both the *direction* and *displacement* of MODEMs preserve information about task transfer relationships. In comparison, LoRA and full fine-tuning exhibit strong correlations in cosine distance, but their Euclidean distances correlate only weakly with task relationships, with coefficients around 0.4 or even negative. Additional results in Table 6 show similar correspondence with task cardinality and data-distribution overlap, further supporting MODEMs as a representation of relational task structure.

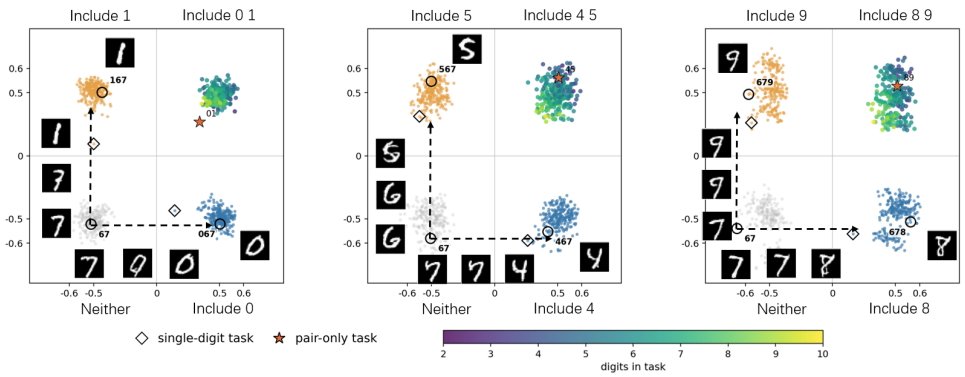


Figure 3: **Semantic structure emerges in the shared MODEM space.** Decomposing learned MODEMs reveals digit-associated directions that capture task-related structure within the representation. The three plots visualize MODEMs in the two-dimensional semantic planes spanned by $(\mathbf{v}_0, \mathbf{v}_1)$, $(\mathbf{v}_4, \mathbf{v}_5)$, and $(\mathbf{v}_8, \mathbf{v}_9)$. In each plane, the embeddings form four distinct groups corresponding to tasks containing neither digit, either one of the two digits, or both digits. Moreover, moving the embedding $\mathbf{m}_{67}$ along the $\mathbf{v}_0$ direction progressively introduces behavior associated with digit 0, illustrating the functional meaning of the learned semantic direction.

3.3 Task Operations: Addition and Subtraction

We next examine whether MODEMs support meaningful algebraic operations. In the controlled MNIST task family, addition and subtraction in model-embedding space produce systematic changes in the generated digit distribution. We use the operations defined in Section 2.3.

Operations in MODEM space can merge or remove knowledge. We evaluate whether addition can construct a triplet task from a pair and a singleton, and whether subtraction can recover a singleton task from a triplet and a pair. A singleton, pair, and triplet denote tasks defined by one, two, and three digits, respectively. As shown in Table 2, addition in MODEM space combines the knowledge encoded by a pair and a singleton to construct a triplet, while subtraction recovers a singleton by retaining its associated knowledge and removing the knowledge contributed by the pair.

Interventions on MODEM components. To understand why these operations work, we examine the semantic organization of the learned representation. MODEMs admit an approximate decomposition into a component shared across tasks and a small set of digit-associated directions:

$$\mathbf{m}_S = \mathbf{m}_{\text{shared}} + \frac{1}{|S|} \sum_{i \in S} \mathbf{v}_i + \boldsymbol{\epsilon}_S, \tag{19}$$

where S is the set of digits defining a task, $\mathbf{v}_i$ denotes the fitted direction associated with digit i, and $\boldsymbol{\epsilon}_S$ captures the remaining task-specific variation.

Empirically, the shared component together with the ten digit-associated directions explains 88.1% of the squared model-update energy, indicating that a large fraction of the learned adaptations is organized according to this simple semantic structure. Figure 3 illustrates the semantic meaning of the digit-associated direction $\mathbf{v}_i$. For example, starting from the MODEM $\mathbf{m}_{67}$, moving along $\mathbf{v}_0$, as in $\mathbf{m}_{67} + \mathbf{v}_0$ or $\mathbf{m}_{67} + 2\mathbf{v}_0$, progressively introduces the behavior associated with digit 0, moving the representation toward that of the $0, 6, 7$ task.

4 Experiments on Vision Tasks

In this section, we focus on the properties of the MODEM space for vision tasks using ViT. Experimental details and additional results on language tasks with GPT-2 are provided in Appendix C. Visualizations of the MODEM spaces for both vision and language tasks are shown in Figure 5.

Table 3: Compared with Full and LoRA, MODEM exhibits a stronger correlation with task relations. Measuring relations by incorporating *MODEM path* further strengthens its correlation with task-space relations.

Task Relation	Metric	Full	LoRA	MODEM	+Path
Transfer Regret	Pearson	-0.43	0.34	0.44	**0.77**
	Spearman	-0.46	0.42	0.49	**0.77**

Table 4: Comparison of addition and subtraction across tuning methods. The pretrained model has 55.3% average accuracy across all tasks. *Specialist* evaluates a model fine-tuned on task A on A; *cross-task* evaluates it on B; and *joint* evaluates a model trained jointly on A and B. *Addition* evaluates the merged adapter $\widehat{AB}$ on either task, while *subtraction* constructs $\widehat{A} = AB - B$, with *retained* A and *residual* B measuring its performance on A and B, respectively.

	Reference			Addition	Subtraction	
Method	**Specialist**	**Cross-task**	**Joint**	$\widehat{AB}\uparrow$	**Retained** $A\uparrow$	**Residual** $B\downarrow$
Full tuning	83.77%	40.58%	83.36%	<u>77.54</u>%	67.26%	**38.83**%
LoRA	83.28%	52.32%	82.1%	75.06%	78.12%	51.99%
MODEM	83.46%	52.78%	83.76%	77.42%	<u>81.57</u>%	65.64%
MODEM + Path	83.46%	52.78%	83.76%	**80.19**%	**82.10**%	<u>51.76</u>%

Experimental setting We use a pretrained CLIP ViT-B/32 model (Radford et al., 2021) as a common initialization and fine-tune it separately on 20 image-classification tasks for evaluation. We compare three adaptation methods. Full fine-tuning updates the vision encoder and visual projection while keeping the text encoder fixed. LoRA learns low-rank updates to the attention block of the vision transformer. MODEM adapts the value projections of the attention block.

Parameter relations and task relations For two different tasks, we measure parameter distance between their learned updates using cosine distance, as defined in (17). We compare these distances against task relation, zero-shot transfer regret, defined in (18). Additionally, our ViT experiments indicate that the condition required by Corollary 2 does not generally hold. Motivated by Theorem 1, we introduce a relation based on the *MODEM path* to account for variations in local functional geometry along the adaptation trajectory. Specifically, given $\mathbf{m}_a$ and $\mathbf{m}_b$, let $\{\Delta\mathbf{m}_a^{(k)}\}_{k=1}^K$ and $\{\Delta\mathbf{m}_b^{(k)}\}_{k=1}^K$ denote their sequence, *i.e.*, *path*, of embedding updates, typically obtained from successive adapter updates during training. We apply the relation using *MODEM path* as

$$\sum_{k=1}^{K} \Phi_a^{(k-1)}(x)\left(\Delta\mathbf{m}_a^{(k)} - \Delta\mathbf{m}_b^{(k)}\right) + \sum_{k=1}^{K}\left[\Phi_a^{(k-1)}(x) - \Phi_b^{(k-1)}(x)\right]\Delta\mathbf{m}_b^{(k)}. \tag{20}$$

MODEM preserves task structure. Table 3 shows the correlation between task relation and MODEM distance. Across all 190 task pairs, MODEM has a higher correlation with task relation than Full or LoRA. Incorporating the MODEM path difference further strengthens the correlation with task-space relations by accounting for the adaptation trajectory and thereby reducing the effect of variations in ViT's local functional geometry.

Operations in MODEM space affect task space. We use the operation defined in Section 2.3. Building on Proposition 2, we introduce *MODEM path merging*:

$$\mathbf{m}_{\text{merge}} = \arg\min_{\mathbf{m}} \left\| \sum_{k=1}^{K} \Phi\left(x; \mathbf{m}^{(k-1)}\right)\Delta\mathbf{m}^{(k)} - \sum_{t\in\mathcal{T}} \lambda_t \sum_{k=1}^{K} \Phi\left(x; \mathbf{m}_t^{(k-1)}\right)\Delta\mathbf{m}_t^{(k)} \right\|_2^2. \tag{21}$$

The results are in Table 4. MODEM addition achieves a mean accuracy of 77.42%. MODEM subtraction retains 81.57% accuracy on the target task while reducing accuracy on the removed task from 83.76% to 65.64%. Incorporating path-aware operations further improves addition to 80.19%, while subtraction retains 82.10% on the target task and reduces removed-task accuracy to 51.76%, a 32.00% reduction from the joint model.

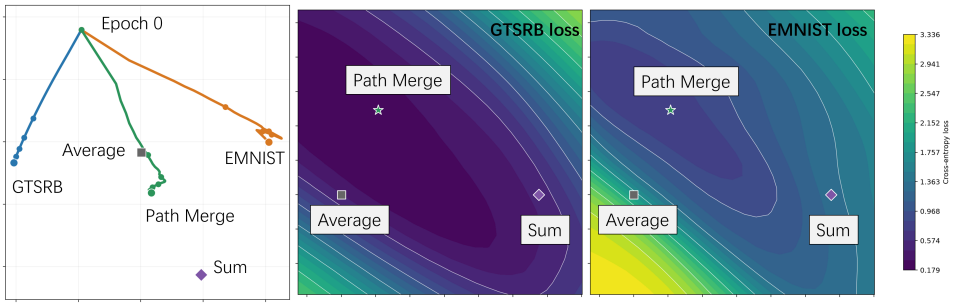


Figure 4: **Illustration of MODEM path merging.** Path merging utilizes the geometry of the MODEM space, leading to a solution with lower loss on both tasks.

5 Related Work

Representations of tasks. Task embeddings summarize properties of learning problems and can support decisions such as selecting a feature extractor. Task2Vec, for example, constructs task representations from the Fisher information of a probe network (Achille et al., 2019). LEEP Nguyen et al. (2020) and LogME You et al. (2021) characterize task relatedness through the transferability of a pretrained representation, estimating how well a source model can support a target task from its predictions or features. Our setting focuses on the adaptations themselves. An embedding both describes an update and decodes into its adapted model.

Parameter-efficient adaptation. Parameter-efficient fine-tuning (PEFT) adapts pretrained models with few trainable parameters, motivated by the low intrinsic dimensionality of adaptation (Houlsby et al., 2019b; Hu et al., 2022; Lester et al., 2021; Aghajanyan et al., 2021). Related work further exploits spectral structure for efficient adaptation and merging (Gargiulo et al., 2025; Han et al., 2023; Zhang & Pilanci, 2024), and studies task composition through parameter arithmetic, sparsification, and adaptive merging (Ilharco et al., 2023; Yadav et al., 2023; Yang et al., 2024; Yu et al., 2024). Our formulation instead learns coordinates within an explicitly specified subspace.

Task arithmetic and composition. Task arithmetic demonstrates addition, negation, and analogies between task vectors (Ilharco et al., 2023). Tangent-space analysis relates successful arithmetic to functional disentanglement (Ortiz-Jimenez et al., 2023). Learned composition methods include LoraHub, which uses a few target examples to compose LoRA modules (Huang et al., 2024); AdaMerging, which learns task-wise or layer-wise coefficients (Yang et al., 2024); and aTLAS, which learns coefficients for parameter blocks (Zhang et al., 2024). FW-Merging studies iterative candidate selection and local merging (Chen et al., 2025). We do not claim arithmetic as new in itself. Instead, we use these operations to assess the proposed representation.

6 Conclusion

We formulate *model embeddings* as representations of task adaptations within a shared subspace induced by a pretrained model. Through theoretical analysis and empirical studies of adaptation fidelity, relational geometry, functional components, and embedding operations, we show how task structure can emerge from the interaction between shared parameter coordinates and their functional effects on data. In conclusion, these results support model embedding space as a useful representation for studying, comparing, and manipulating learned task adaptations.

Discussion and limitations. Our study primarily uses controlled experiments to establish model embedding space as a representation of task adaptations. While we extend evaluation to vision tasks with ViT and language tasks with GPT-2, generalization to larger models and more diverse real-world tasks remains to be explored.

7 AI DISCLOSURE

In this work, we used generative AI tools for providing feedback on research methodology or experiments, and to assist in the writing of proofs. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, generate synthetic data sets, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results are not applicable to this work. Additionally, we used generative AI tools for brainstorming, sourcing/searching for information, and editing a research paper to improve readability. We have reviewed all AI-assisted work. We have checked the proof (assumptions, inequalities, etc.) used by AI. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, 2019.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL-IJCNLP*, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *PAMI*, 2013.

Hao Mark Chen, Shell Xu Hu, Wayne Luk, Timothy Hospedales, and Hongxiang Fan. Fw-merging: Scaling model merging with frank-wolfe optimization. In *ICCV*, 2025.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *CVPR*, 2025.

Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. In *ICCV*, 2023.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019a.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *COLM*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2013.

Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *ICML*, 2020.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *NeurIPS*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024.

Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Fangzhao Zhang and Mert Pilanci. Spectral adapter: Fine-tuning in spectral space. *NeurIPS*, 2024.

Frederic Z Zhang, Paul Albert, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. Knowledge composition using task vectors with learned anisotropic scaling. *NeurIPS*, 2024.

A Appendix

A.1 Model Embeddings for Attention

MODEM for attention block. Consider an attention layer with H attention heads and model dimension d. Let

$$d_h = \frac{d}{H} \tag{22}$$

be the output dimension of each head. We write the pretrained value projection as a row-wise partition

$$\mathbf{W}_{0,v} = \begin{bmatrix} \mathbf{W}_{0,v}^{(1)} \\ \vdots \\ \mathbf{W}_{0,v}^{(H)} \end{bmatrix}, \qquad \mathbf{W}_{0,v}^{(h)} \in \mathbb{R}^{d_h \times d}. \tag{23}$$

For task t, each attention head is associated with a model-embedding coordinate matrix

$$\mathbf{M}_t^{(h)} \in \mathbb{R}^{d_h \times d_h}. \tag{24}$$

Using the pretrained value projection as a fixed basis, we decode this coordinate matrix as

$$\Delta \mathbf{W}_{t,v}^{(h)} = \mathbf{M}_t^{(h)\top} \mathbf{W}_{0,v}^{(h)}. \tag{25}$$

The complete value-projection update is obtained by concatenating the head-wise updates,

$$\Delta \mathbf{W}_{t,v} = \begin{bmatrix} \Delta \mathbf{W}_{t,v}^{(1)} \\ \vdots \\ \Delta \mathbf{W}_{t,v}^{(H)} \end{bmatrix}. \tag{26}$$

For an input representation $\mathbf{x}$, the adapted value projection is therefore

$$\begin{aligned} \mathbf{v}_t &= \mathbf{x} \left(\mathbf{W}_{0,v} + \Delta \mathbf{W}_{t,v} \right)^\top \\ &= \mathbf{x} \mathbf{W}_{0,v}^\top + \mathbf{x} \Delta \mathbf{W}_{t,v}^\top. \end{aligned} \tag{27}$$

In our implementation, the query, key, and output projections $\mathbf{W}_{0,q}$, $\mathbf{W}_{0,k}$, and $\mathbf{W}_{0,o}$ are frozen, and only the value path is adapted. Consequently, the attention scores continue to be determined by the pretrained query and key projections, whereas the task-specific model embedding modifies the values aggregated by these attention weights.

The interpretation of h depends on the model component: for an MLP weight, h indexes a block of selected singular directions, whereas for an attention value projection, h indexes a pretrained attention head. These components share the same abstract role, each $\mathbf{M}_t^{(h)}$ provides coordinates for a fixed linear decoder.

B Details of the Controlled MNIST Study

B.1 Task family and experimental protocol

The controlled task family consists of uniform distributions over all nonempty digit subsets rather than arbitrary nonuniform mixtures. For $S \subseteq \{0, \ldots, 9\}$,

$$P_{t_S}(x) = \frac{1}{|S|} \sum_{k \in S} P_k(x), \qquad q_k(S) = \frac{1[k \in S]}{|S|}. \tag{28}$$

The complete suite therefore contains $2^{10} - 1 = 1023$ tasks. The experiment studies changes in digit support and equal-proportion compositions; it does not test arbitrary nonuniform mixture weights.

The generative backbone is a rectified-flow transformer for 28×28 grayscale images. Images are divided into 4×4 patches. The transformer has hidden width 256, depth 3, eight attention heads,

Table 5: Summary of notation.

Notation	Definition	Role
P_t	Data distribution associated with task t	Defines task data
ℓ_t	Per-sample loss for task t	Defines the task objective
$\mathcal{L}_t(\Delta\mathbf{W})$	Population objective $\mathbb{E}_{z\sim P_t}\left[\ell_t(f_{\mathbf{W}_0+\Delta\mathbf{W}}; z)\right]$	Defines the learning problem
S_t	Finite training set sampled from task t	Used for empirical adaptation
$\widehat{\mathcal{L}}_t(\mathbf{m})$	Empirical objective evaluated through $\mathbf{W}(\mathbf{m}) = \mathbf{W}_0 + D(\mathbf{m})$	Used to learn $\mathbf{m}_t$
$\mathbf{W}_0$	Parameters of the common pretrained model	Shared reference point
$\Delta\mathbf{W}_t$	Learned parameter update for task t	Adaptation in the ambient parameter space
D	Fixed decoder mapping model-embedding coordinates to parameter updates	Defines the structured parameterization
$\mathbf{m}_t$	Learned model embedding for task t	Task representation
$\mathbf{M}_t^{(h)}$	Coordinates of task t associated with component or block h	Component-level representation
$\mathcal{E}_{\mathbf{W}_0}$	Structured update subspace $\{D(\mathbf{m}) : \mathbf{m} \in \mathcal{C}\}$	Subspace of the ambient parameter-update space
$\mathbf{W}(\mathbf{m})$	Parameters induced by embedding $\mathbf{m}$, $\mathbf{W}(\mathbf{m}) = \mathbf{W}_0 + D(\mathbf{m})$	Embedding-parameterized model family
$f_{\mathbf{W}}(x)$	Model function parameterized by arbitrary weights $\mathbf{W}$	Generic model function
$f(x;\mathbf{m})$	Model function induced by embedding $\mathbf{m}$, $f(x;\mathbf{m}) = f_{\mathbf{W}(\mathbf{m})}(x)$	Function expressed in embedding coordinates
$f_0(x)$	Pretrained function, $f_0(x) = f(x;\mathbf{0}) = f_{\mathbf{W}_0}(x)$	Functional reference point
$f_t(x)$	Learned function for task t, $f_t(x) = f(x;\mathbf{m}_t)$	Functional realization of task t
$\Phi(x;\mathbf{m})$	Functional Jacobian $\frac{\partial F(x;\mathbf{m})}{\partial \mathbf{m}}$	Local functional response to embedding coordinates
$\bar{\Phi}_t(x)$	Functional map averaged along the path from $\mathbf{0}$ to $\mathbf{m}_t$	Connects embedding displacement to functional change
$\mathcal{A}$	Adaptation procedure mapping task data to a model embedding	Learning algorithm

MLP ratio 4, and no dropout. For data sample x_1, Gaussian noise x_0, and $t \sim \mathrm{Unif}[0,1]$, training minimizes

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{x_1,x_0,t}\left[\left\|v_\theta\big((1-t)x_0 + tx_1, t\big) - (x_1 - x_0)\right\|_2^2\right]. \tag{29}$$

Sampling uses 100-step Heun integration.

The backbone is pretrained for 1000 epochs on the EMNIST `letters` training split. EMNIST glyphs are rotated and horizontally flipped to match the MNIST orientation. Pretraining uses batch size 512, AdamW with learning rate 2×10^{-4}, weight decay 10^{-4}, and seed 0.

For each downstream task t_S, a model embedding $\mathbf{m}_{t_S}$ is trained independently on the corresponding class-balanced MNIST subset for 500 epochs. All available MNIST training samples from the selected digit classes are used. Fine-tuning uses AdamW with learning rate 10^{-3} and weight decay 10^{-4}.

In the primary dense experiment, only the attention value projections are adapted; query, key, output, and the remaining backbone parameters stay frozen. The resulting dense representation contains 24,576 active model-embedding coordinates.

Table 6: Correlation between task distances and attention-parameter distances.

		Euclidean distance		Cosine distance	
Task distance	Method	Pearson	Spearman	Pearson r	Spearman ρ
Jaccard	Model embedding	0.873	0.865	0.928	0.937
	LoRA	-0.178	-0.194	0.879	0.894
	Full tuning	0.655	0.642	0.936	0.941
MMD	Model embedding	0.668	0.726	0.912	0.916
	LoRA	-0.490	-0.440	0.808	0.867
	Full tuning	0.359	0.427	0.913	0.924
Transfer	Model embedding	0.741	0.826	0.945	0.962
	LoRA	-0.372	-0.315	0.840	0.921
	Full tuning	0.481	0.569	0.949	0.970

B.2 Task relation metrics

Digit-set overlap. Jaccard distance measures the fraction of digits absent from the intersection relative to the union:

$$d_{\mathrm{Jac}}(i,j) = 1 - \frac{|S_i \cap S_j|}{|S_i \cup S_j|}. \tag{30}$$

It is zero for identical digit sets and one for disjoint sets.

Image-distribution discrepancy. We estimated maximum mean discrepancy (MMD) from real MNIST training images. A fixed classifier produced a 128-dimensional feature vector for each of the 60,000 images. Features were standardized using the training-set mean and standard deviation. We approximated an equal-weight mixture of three Gaussian kernels with bandwidths $0.5m$, m, and $2m$, where m was the median nonzero distance among 2,048 sampled standardized feature vectors. Each kernel used 1,024 random Fourier features, with fixed projections shared by all images and tasks. If $\boldsymbol{\mu}_d$ is the mean random-feature vector for digit d, the task mean and distance are

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{d \in S_i} \boldsymbol{\mu}_d, \tag{31}$$

$$D_{\mathrm{MMD}}(i,j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2. \tag{32}$$

This is an approximation to MMD, rather than squared MMD.

B.3 Visualization of Adapter Manifolds During Fine-Tuning

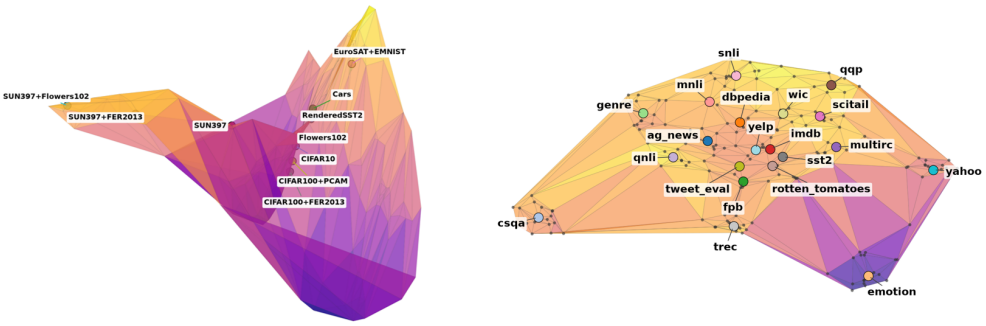
We studied how task-specific adapters evolve during MNIST fine-tuning. This is the process of creating visualization for Figure 1. Starting from an EMNIST-letter-pretrained transformer with three attention-adapter layers and no trainable output head, we fine-tuned one adapter for every non-empty subset of the ten MNIST digits. This produced $2^{10} - 1 = 1023$ task adapters. Adapter snapshots were saved every 50 epochs from epoch 50 through epoch 500.

For each task t and epoch e, we formed an adapter vector $\mathbf{a}_{t,e}$ by concatenating all trainable adapter coefficients across the transformer layers; output-head parameters were excluded. To make coordinates comparable over training, we did *not* fit PCA independently at each epoch. Instead, we fit one centered PCA basis to the union of adapter vectors from all selected epochs:

$$\bar{\mathbf{a}} = \frac{1}{|\mathcal{E}||\mathcal{T}|} \sum_{e \in \mathcal{E}} \sum_{t \in \mathcal{T}} \mathbf{a}_{t,e}, \qquad \mathbf{z}_{t,e} = \mathbf{W}^\top (\mathbf{a}_{t,e} - \bar{\mathbf{a}}),$$

where $\mathcal{E} = \{50, 100, \ldots, 500\}$, $\mathcal{T}$ denotes the 1023 digit-combination tasks, and $\mathbf{W}$ contains the first three principal directions. We used a randomized SVD implementation to fit this common basis without materializing an impractically large joint coefficient matrix.

Each epoch is visualized separately in the common $(\mathrm{PC}_1, \mathrm{PC}_2, \mathrm{PC}_3)$ coordinate system with identical axis limits. Consequently, apparent movement of a point across figures corresponds to a change



(a) Vision Tasks (b) Language Tasks

Figure 5: Visualization of MODEM space.

in its adapter coefficients, rather than to an arbitrary rotation, reflection, or rescaling induced by re-fitting a dimensionality-reduction method. Single-digit adapters act as labeled anchors, while multi-digit tasks are colored by mixtures of their constituent digit colors.

For visualization only, we render a translucent surface over the projected points. Surface faces and visible edges are determined by local neighborhoods in the shared $(\mathrm{PC}_1, \mathrm{PC}_2)$ plane, avoiding spurious long-range connections caused by computing neighbors in the original high-dimensional adapter space. The displayed height is a globally normalized PC_3 coordinate with optional local smoothing.

C Additional Experimental Details

C.1 Vision Tasks

We evaluate on 20 vision datasets: Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN, CIFAR-100, STL-10, Flowers-102, Oxford-IIIT Pet, PCAM, FER2013, EMNIST, CIFAR-10, Food-101, Fashion-MNIST, rendered SST-2, and KMNIST. For each task, we use a stratified 10% subset of the training set, together with its task-specific class labels and text-prompt classifier, and report accuracy on the clean test split.

All methods are initialized from the same pretrained model and trained with cross-entropy loss using AdamW; checkpoints are selected by validation accuracy. For parameter-space comparisons, we use the effective LoRA update rather than its non-unique factors, and restrict full fine-tuning updates to the value projections so that all methods are compared over the same parameter scope.

C.2 Language Tasks

We evaluate on 20 NLP datasets: TREC, Emotion, CommonsenseQA, Financial PhraseBank, MNLI, SNLI, TweetEval, QQP, MNLI Genre, SciTail, Yahoo Answers, MultiRC, WiC, QNLI, Yelp, AG News, Rotten Tomatoes, SST-2, DBpedia, and IMDB, and construct all 190 pairwise joint-task models. For each task, we use 500 training examples together with task-specific label words and prompt templates.

Experimental setting We use a small pretrained GPT-2 model as a base model and evaluate 20 text classification tasks. We compare three adaptation methods. Full fine-tuning updates all parameters within the value blocks and MLP matrices of the transformer. LoRA learns a low-rank update to the value blocks and MLP matrices of the transformer. Model embedding adapts value blocks and MLP.

For MODEM, each of the 12 attention heads in each of the 12 Transformer layers uses a separate 64×64 update. Each MLP matrix additionally uses a block-diagonal update consisting of four 8×8 blocks within its rank-32 pretrained SVD basis, resulting in 595,968 trainable parameters in total.

Table 7: Comparison of adaptation capability across tuning methods.

	Metric	Full	LoRA	Embedding
Reference	Specialist	76.69%	75.60%	76.21%
	Cross-task	47.44%	45.59%	48.09%
	Joint AB	76.79%	76.05%	76.15%

All methods are initialized from the same pretrained model and trained for 50 epochs using masked cross-entropy loss on the verbalized label tokens and AdamW; checkpoints are selected by validation accuracy. For parameter-space comparisons, we follow the same protocol as in the vision experiments, restricting comparisons to the value projections and MLP matrices and using the effective LoRA updates rather than their factorized parameters.

Parameter relations and task relations Across all 210 task pairs, model-embedding cosine distance is associated with GPT-2-feature Fréchet distance (Spearman $\rho = 0.473$), more strongly than rank-32 LoRA ($\rho = 0.424$) or full fine-tuning ($\rho = 0.381$).

Adaptation in model embedding space. The results are in Table 4. MODEM achieves comparable accuracy to Full and LoRA.

D Properties of Model Embedding Representations

We now study why the model embedding introduced above can serve as a representation of a downstream task. Recall that a task t induces a learned model embedding $\mathbf{m}_t$, which is decoded into the corresponding parameter adaptation

$$\Delta \mathbf{W}_t = D(\mathbf{m}_t). \tag{33}$$

The resulting task-specific model is

$$f_t(x) := f_{\mathbf{W}_0 + D(\mathbf{m}_t)}(x). \tag{34}$$

Thus, the model embedding provides a common coordinate representation of the functional change learned from each downstream task. We study three properties of this representation: its relation to task behavior, its geometry across tasks, and its decomposition into individual coordinate components.

Assumption 1. *Assume that $f(x; \mathbf{m})$ is differentiable with respect to $\mathbf{m}$ and that its functional Jacobian*

$$\Phi(x; \mathbf{m}) := \frac{\partial f(x; \mathbf{m})}{\partial \mathbf{m}}$$

is $L(x)$-Lipschitz along the path from $\mathbf{0}$ to $\mathbf{m}_t$.

Proposition 2. *Let*

$$\mathbf{m}_t^{(k)} := \alpha_k \mathbf{m}_t, \qquad \Delta \mathbf{m}_t^{(k)} := \mathbf{m}_t^{(k)} - \mathbf{m}_t^{(k-1)},$$

where $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_K = 1$. Then

$$f(x; \mathbf{m}_t) - f_0(x) = \sum_{k=1}^{K} \Phi\Big(x; \mathbf{m}_t^{(k-1)}\Big)\, \Delta \mathbf{m}_t^{(k)} + R_t(x), \tag{35}$$

where

$$\|R_t(x)\| \le \frac{L(x)}{2} \sum_{k=1}^{K} \|\Delta \mathbf{m}_t^{(k)}\|_2^2. \tag{36}$$

Here $f_0(x) = f(x; \mathbf{0})$ denotes the pretrained model.

The functional change associated with task t can therefore be represented by the accumulation of local linear changes along the path from the pretrained model $\mathbf{0}$ to its MODEM $\mathbf{m}_t$. Define the

path-averaged functional map

$$\bar{\Phi}_t(x) := \sum_{k=1}^{K} (\alpha_k - \alpha_{k-1}) \Phi\Big(x; \mathbf{m}_t^{(k-1)}\Big). \tag{37}$$

Since $\Delta \mathbf{m}_t^{(k)} = (\alpha_k - \alpha_{k-1})\mathbf{m}_t$, Proposition 2 gives

$$f(x; \mathbf{m}_t) - f_0(x) = \bar{\Phi}_t(x)\mathbf{m}_t + R_t(x) \approx \bar{\Phi}_t(x)\mathbf{m}_t. \tag{38}$$

Proof. For each path segment, the fundamental theorem of calculus gives

$$\begin{aligned} &f(x; \mathbf{m}_t^{(k)}) - f(x; \mathbf{m}_t^{(k-1)}) \\ &\quad = \int_0^1 \Phi\Big(x; \mathbf{m}_t^{(k-1)} + s\Delta \mathbf{m}_t^{(k)}\Big) \Delta \mathbf{m}_t^{(k)} \, ds. \end{aligned} \tag{39}$$

Subtracting the Jacobian at the beginning of the segment and using its $L(x)$-Lipschitz continuity yields

$$\Big\| f(x; \mathbf{m}_t^{(k)}) - f(x; \mathbf{m}_t^{(k-1)}) - \Phi(x; \mathbf{m}_t^{(k-1)}) \Delta \mathbf{m}_t^{(k)} \Big\| \leq \frac{L(x)}{2} \|\Delta \mathbf{m}_t^{(k)}\|_2^2. \tag{40}$$

Summing over the K segments telescopes from $\mathbf{0}$ to $\mathbf{m}_t$, which gives (35) and (36). □

Theorem 1. *For two tasks t_i and t_j,*

$$\begin{aligned} f(x; \mathbf{m}_i) - f(x; \mathbf{m}_j) = &\underbrace{\bar{\Phi}_i(x)(\mathbf{m}_i - \mathbf{m}_j)}_{\textit{MODEM difference}} \\ &+ \underbrace{\big[\bar{\Phi}_i(x) - \bar{\Phi}_j(x)\big]\mathbf{m}_j}_{\textit{functional geometry mismatch}} + R_{ij}(x), \end{aligned} \tag{41}$$

where $R_{ij}(x) := R_i(x) - R_j(x)$. Thus, up to local linearization error, functional differences between tasks are determined by both their MODEM differences and the mismatch between the functional geometries traversed by their adaptation paths.

Proof. Applying (38) to the two tasks and subtracting gives

$$f(x; \mathbf{m}_i) - f(x; \mathbf{m}_j) = \bar{\Phi}_i(x)\mathbf{m}_i - \bar{\Phi}_j(x)\mathbf{m}_j + R_{ij}(x).$$

Adding and subtracting $\bar{\Phi}_i(x)\mathbf{m}_j$ directly yields (41). □

Corollary 2. *If the two tasks traverse similar functional geometry,*

$$\bar{\Phi}_i(x) \approx \bar{\Phi}_j(x) \approx \bar{\Phi}(x),$$

and the local linearization errors are small, then

$$f(x; \mathbf{m}_i) - f(x; \mathbf{m}_j) \approx \bar{\Phi}(x)(\mathbf{m}_i - \mathbf{m}_j). \tag{42}$$

If the adaptation paths additionally remain close to the pretrained functional geometry, $\bar{\Phi}(x) \approx \Phi_0(x)$, then

$$f(x; \mathbf{m}_i) - f(x; \mathbf{m}_j) \approx \Phi_0(x)(\mathbf{m}_i - \mathbf{m}_j). \tag{43}$$

Proof. Under shared local geometry, the functional-geometry mismatch term in (41) becomes negligible, yielding (42). Substituting $\bar{\Phi}(x) \approx \Phi_0(x)$ gives (43). □

Hence, when tasks traverse similar local functional geometry, differences between their MODEMs provide coordinates for the corresponding functional differences under an approximately shared functional map.