MERGODIC: ALMOST-LOSSLESS MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning a shared pretrained model yields a rapidly growing population of task-specific experts, and model merging has emerged as the standard way to consolidate them into a single deployable artifact without retraining. Existing approaches face a fundamental tension. Static merging methods collapse all experts into one set of weights and are compact, but parameter interference causes accuracy to degrade as the number of tasks grows. Task-conditioned methods, often implemented with routing mechanisms, recover much of this lost accuracy by retaining task-specific signal, but they reintroduce exactly what merging set out to remove: per-task masks, adapters, or expert modules whose storage scales with the number of tasks. We ask whether a single shared representation can instead regenerate any requested expert on demand. We answer in the affirmative with Mergodic, a task-conditioned merging method grounded in a classical result from ergodic theory: by the Kronecker-Weyl equidistribution theorem, the orbit of a one-dimensional irrational rotation is dense in the n-dimensional torus, so the n cross-task values at any parameter coordinate can be approximated by a single scalar phase. Mergodic discretizes this phase into one integer code per coordinate—shared across all experts and independent of the number of tasks—and reconstructs each expert through a deterministic, task-conditioned rotation map. The result is an almost-lossless merge: experts are recovered with negligible reconstruction error at a fraction of the storage of the original collection. We evaluate Mergodic under both expert-availability regimes: offline merging, where all experts are available simultaneously, and continual merging, where experts arrive sequentially and previous checkpoints are discarded. Across vision and natural language benchmarks, Mergodic attains state-of-the-art performance while remaining highly compact.

1 INTRODUCTION

The dominant recipe in modern machine learning is to fine-tune a single pretrained model into many specialized experts, one per task or domain. This recipe is enormously productive, but it leaves practitioners with a growing inventory of checkpoints that share an ancestor yet must each be stored, versioned, and served independently. *Model merging* addresses this burden directly: it fuses several fine-tuned models into a single artifact, in weight space and without access to the original training data, so that the combined capabilities can be deployed at the cost of (close to) one model (Wortsman et al., 2022; Ilharco et al., 2023; Yang et al., 2024a).

A common abstraction for merging is the *task vector*—the difference between a fine-tuned model and its pretrained initialization—which isolates the knowledge acquired during fine-tuning as a direction in weight space (Ilharco et al., 2023). Task vectors can be added, scaled, and negated, and simple arithmetic over them already yields competitive multi-task models. However, summing or averaging task vectors is intrinsically lossy. Different experts disagree in both magnitude and sign at the same coordinates, and these conflicts cause influential updates to cancel (Yadav et al., 2025); the gap between a merged model and its fine-tuned sources widens as more experts are combined (Wang et al., 2025b). A substantial line of work mitigates this interference through sparsification and conflict resolution—trimming low-magnitude updates and electing a consensus sign (Yadav et al., 2023), randomly dropping and rescaling task vectors (Yu et al., 2024), or weighting parameters by estimated importance (Matena & Raffel, 2022; Jin et al., 2023). These methods push the frontier of *static* merging, in which the output is one fixed set of weights, but they cannot escape its central limitation: a single point in weight space must serve every task at once, so some accuracy is always surrendered.

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.

A second, increasingly successful paradigm sidesteps this compromise by making the merge *task-conditioned*. Rather than committing to one static solution, these methods retain task-specific information and select or combine it dynamically—either per task or per input—using a router or gating mechanism (Huang et al., 2024; Du & Lin, 2026; Zheng & Wang, 2025; Tang et al., 2024b;a). By separating shared knowledge from exclusive, task-specific knowledge and recombining them on demand, they recover much of the performance lost to interference and close the gap to the fine-tuned experts. This accuracy, however, comes at a price that runs counter to the original motivation for merging. Task-conditioned methods store explicit per-task structure—masks, low-rank experts, adapters, or compressed exclusive vectors—and this storage grows with the number of tasks. In effect, the field has traded the memory savings of merging for the accuracy of conditioning.

This motivates the central question of our work: *Can the information in many task vectors be encoded into a single, compact shared representation, from which the task vector for any requested task is reconstructed on demand—without storing per-task modules whose size grows with the number of tasks?*

We answer this question by drawing on a classical phenomenon from ergodic theory (Cornfeld et al., 2012). The Kronecker-Weyl equidistribution theorem states that when the frequencies $a_1, \ldots, a_n$ are linearly independent over the rationals, the orbit of the one-dimensional flow $\theta \mapsto \big(\rho(\theta a_1), \ldots, \rho(\theta a_n)\big)$, where ρ denotes the fractional part, is dense in the unit hypercube $[0,1]^n$ (Weyl, 1916; Kuipers & Niederreiter, 1974). Geometrically, a single scalar travelling along an irrational-rotation trajectory passes arbitrarily close to *every* point of the n-torus. The implication for merging is direct. Consider a fixed parameter coordinate and collect, across the n experts, the n values that the task vectors take there; after normalization, these form a single point in $[0,1]^n$. The density theorem says this n-dimensional cross-task point can be approximated by one scalar phase θ. Discretizing θ over a finite search budget turns it into a compact integer code, and the value for any individual task t is recovered simply by evaluating the deterministic map $\rho(\theta a_t)$ at that task's frequency.

Building on this observation, we introduce *Mergodic*, a task-conditioned merging method whose stored representation is, per parameter coordinate, a single integer that is shared across all experts and whose count does not grow with the number of tasks. Given a target task index—either directly by user, or a lightweight router—Mergodic reconstructs the corresponding expert by applying an irrational-rotation map, followed by a denormalization. Because each coordinate of each expert is regenerated from the shared code rather than averaged into a compromise, the reconstruction is *almost-lossless*: experts are recovered with negligible error while the collection is stored at a fraction of its original size. Mergodic thus occupies a previously empty corner of the design space—task-conditioned like router-based methods, yet nearly as compact as static merging. We provide a broader discussion of related work in Appendix A.2.

Our contributions are as follows:

- We formulate model merging as an encode-decode problem: a compact shared representation should preserve the information of multiple experts while enabling task-specific models to be recovered on demand.
- We propose *Mergodic*, which uses irrational rotations to encode the cross-task values at each parameter coordinate into a single finite integer code. We show that its reconstruction is asymptotically lossless and characterize its finite-budget controllable storage–accuracy trade-off. Its also admits a highly parallel and efficient GPU implementation.
- We show that Mergodic achieves almost-lossless reconstruction and state-of-the-art performance across vision and natural language benchmarks in both offline and continual merging settings, while requiring substantially less storage than retaining all the original experts.

The remainder of the paper is organized as follows. Section 2 introduces the problem formulation and mathematical preliminaries underlying our approach. Section 3 presents Mergodic and its encoding and task-conditioned decoding procedures. Section 4 presents our experimental evaluation.

2 Preliminaries and Problem Formulation

This section establishes the formulation and mathematical foundations used by Mergodic. We first formalize task-conditioned model merging as an encoder-decoder problem over task-specific experts

and their task vectors. We then introduce the irrational-rotation construction underlying Mergodic, beginning with the Kronecker-Weyl density theorem and subsequently adapting this idealized result to a practical finite integer encoding scheme.

2.1 TASK-CONDITIONED MODEL MERGING

Let f^{pre} denote a pretrained model, and let $\mathcal{F} = \{f^{(1)}, \ldots, f^{(n)}\}$ denote n expert models obtained by independently fine-tuning f^{pre} on n different tasks. For task t, let $\mathcal{V}_t$ denote a held-out evaluation set and let $\mathrm{Perf}(f, \mathcal{V}_t)$ denote the corresponding task-performance metric, such as accuracy, F1 score, or another task-specific metric for model f over $\mathcal{V}_t$.

We consider *task-conditioned model merging*, where the collection of experts is consolidated into a compact shared representation from which task-specific models can be produced on demand. Formally, a task-conditioned merging method consists of an encoding function

$$\Phi = \mathcal{E}(\mathcal{F}),$$

which maps the collection of experts to a shared merged representation Φ, together with a task-conditioned decoding function

$$\widehat{f}^{(t)} = \mathcal{D}(\Phi, t), \qquad t \in \{1, \ldots, n\},$$

which reconstructs the model used for task t during inference. The target task index may be provided explicitly through task metadata or inferred from the input using a lightweight router. The goal is to preserve the capabilities of the original experts while substantially reducing the storage required to retain them. Accordingly, we seek an encoder-decoder pair $(\mathcal{E}, \mathcal{D})$ that maximizes the average downstream performance of the decoded task-specific models under a prescribed storage budget:

$$\max_{\mathcal{E},\mathcal{D}} \quad \frac{1}{n}\sum_{t=1}^{n} \mathrm{Perf}\left(\mathcal{D}(\mathcal{E}(\mathcal{F}), t), \mathcal{V}_t\right) \quad \text{s.t.} \quad \mathrm{Storage}(\Phi) \le B.$$

Here, B denotes the storage budget for the merged representation, which is typically chosen such that, $B \ll \sum_{t=1}^{n} \mathrm{Storage}\big(f^{(t)}\big)$. In addition to satisfying this storage constraint, the decoder should support fast and efficient task-conditioned model recovery. Thus, the desired representation should jointly provide high task performance, compact storage, and inexpensive recovery of the model associated with a requested task.

This formulation also subsumes conventional *static* model merging as a special case. In static merging, the merged representation itself is the deployed model and the decoder is independent of the target task: $\mathcal{D}(\Phi, t) = \Phi,\ \forall t$, so a single fixed set of weights must serve all tasks. Task-conditioned methods instead allow the deployed weights to depend on t, thereby preserving task-specific information that may otherwise be lost when all experts are collapsed into a single static solution.

Since all experts originate from the same pretrained initialization, their task-specific changes can be conveniently represented through *task vectors*. For simplicity of notation, we formulate the following definitions for a single parameter matrix and apply the same construction independently across model layers. Let $\boldsymbol{W}^{\mathrm{pre}} \in \mathbb{R}^{m\times d}$ denote a parameter matrix of the pretrained model, and let $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots, \boldsymbol{W}^{(n)}$ denote the corresponding matrices of the n fine-tuned experts. The task vector for task t is defined as $\boldsymbol{\tau}^{(t)} = \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\mathrm{pre}}, \quad t \in \{1, \ldots, n\}$. We denote the collection of task-vector matrices by $\mathcal{T} = \{\boldsymbol{\tau}^{(1)}, \ldots, \boldsymbol{\tau}^{(n)}\}$.

Under the task-conditioned formulation, the merged representation Φ must retain sufficient information about $\mathcal{T}$ to produce, for a requested task t, a task-specific approximation $\widehat{\boldsymbol{\tau}}^{(t)}$ and hence the corresponding parameter matrix $\widehat{\boldsymbol{W}}^{(t)} = \boldsymbol{W}^{\mathrm{pre}} + \widehat{\boldsymbol{\tau}}^{(t)}$. Applying this procedure to all parameter matrices yields the full decoded model $\widehat{f}^{(t)}$.

Mergodic instantiates this paradigm by constructing a compact shared representation Φ from which each task vector can be reconstructed with high fidelity. Rather than treating reconstruction error as a separate definition of the model-merging objective, we use faithful task-vector reconstruction as the mechanism through which Mergodic seeks to preserve the downstream performance of the original experts. We develop this construction after introducing the mathematical preliminaries below.

2.2 Irrational Rotations and the Kronecker-Weyl Theorem

Our method is motivated by a classical density result for the orbit of an irrational rotation. Let $\rho(z) = z - \lfloor z \rfloor$ denote the fractional-part function, mapping a real number to $[0, 1)$. The result we use is the Kronecker-Weyl equidistribution theorem; we state the density consequence we need and defer the proof sketch to Appendix A.5.

Theorem 1 (Kronecker-Weyl; density of the linear torus flow)**.** *Let $a_1, \ldots, a_n \in \mathbb{R}$ be* linearly independent over the rationals $\mathbb{Q}$ *(equivalently,* rationally independent*): the only rational numbers $h_1, \ldots, h_n$ with $h_1 a_1 + \cdots + h_n a_n = 0$ are $h_1 = \cdots = h_n = 0$. Then the orbit*

$$\Big\{ \big(\rho(\theta a_1), \rho(\theta a_2), \ldots, \rho(\theta a_n)\big) : \theta \in [0, \infty) \Big\}$$

is dense in $[0, 1]^n$. Equivalently, for any target $\boldsymbol{w} = (w_1, \ldots, w_n) \in [0, 1]^n$ and any $\epsilon > 0$, there exists a scalar $\theta^\star \in [0, \infty)$ such that

$$|w_t - \rho(\theta^\star a_t)| < \epsilon, \qquad t = 1, \ldots, n.$$

The theorem says the one-dimensional trajectory $\theta \mapsto (\rho(\theta a_1), \ldots, \rho(\theta a_n))$ visits every neighborhood of every point in the unit hypercube $[0, 1]^n$.

Mergodic applies this observation coordinate-wise to task-vector matrices. At a fixed coordinate (i, j), the values of that coordinate across the n task vectors form an n-dimensional vector. After mapping these values to $[0, 1]$, the density theorem suggests that this cross-task coordinate vector can be approximated by a single scalar evaluated along an irrational-rotation orbit.

2.3 From Dense Orbits to Finite Integer Codes

Theorem 1 shows that, given rationally independent coefficients $a_1, \ldots, a_n$, the one-dimensional orbit $\theta \mapsto (\rho(\theta a_1), \rho(\theta a_2), \ldots, \rho(\theta a_n))$ comes arbitrarily close to every point in $[0, 1]^n$. In particular, an n-dimensional target point can, in principle, be approximated using only the scalar θ. This existence result assumes, however, that θ may take any nonnegative real value and may grow without bound. A practical encoding scheme must instead use a finite set of representable codes. We therefore restrict θ to the integers $\theta \in \{0, 1, \ldots, K\}$, where K is a finite search budget. For a target point $\boldsymbol{w} = (w_1, \ldots, w_n) \in [0, 1]^n$, we select the integer whose orbit point most closely approximates $\boldsymbol{w}$:

$$\theta^\star = \arg \min_{\theta \in \{0, \ldots, K\}} \sum_{t=1}^{n} \left(w_t - \rho(\theta a_t)\right)^2 .$$

The scalar $\theta^\star$ then acts as a finite integer code for the entire target point, whose coordinates are approximately recovered as

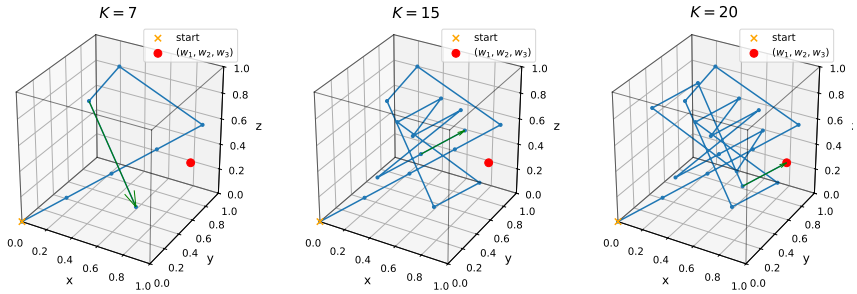
$$\widehat{w}_t = \rho(\theta^\star a_t), \qquad t = 1, \ldots, n.$$

Figure 1 visualizes this construction in three dimensions. For this illustration, we choose $a_1 = \frac{1}{\pi+1}$, $a_2 = \frac{1}{\pi+2}$, $a_3 = \frac{1}{\pi+3}$, which define the orbit

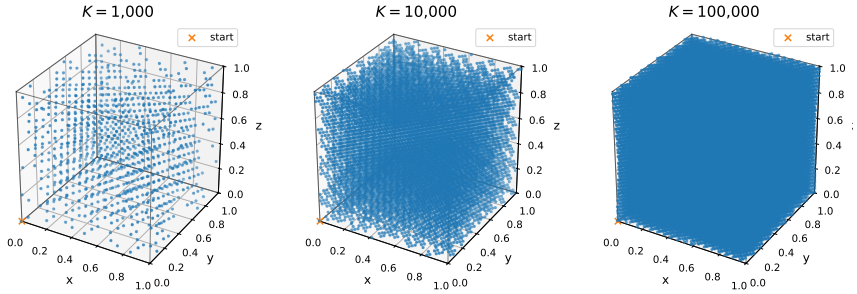
$$\gamma(\theta) = \left(\rho\left(\frac{\theta}{\pi+1}\right), \rho\left(\frac{\theta}{\pi+2}\right), \rho\left(\frac{\theta}{\pi+3}\right) \right).$$

In Figure 1(a), three numbers in $[0, 1]$ form a target point $\boldsymbol{w} = (w_1, w_2, w_3) \in [0, 1]^3$. Each integer value of θ produces one candidate point along the orbit. As the search budget increases, more candidates are considered, and a point closer to the target may become available. Figure 1(b) shows this effect at a larger scale: increasing K makes the finite collection of orbit points progressively denser throughout the unit cube.

For any fixed K, the finite candidate set does not inherit the arbitrarily accurate approximation guarantee of the infinite orbit. Nevertheless, its best approximation error is non-increasing with K, because increasing the search budget preserves all previously available candidates while adding new ones. The finite construction therefore exposes a direct trade-off between encoding cost and approximation quality. We analyze this trade-off formally in Appendix A.8, where we provide a finite-budget lower and upper bound on the worst-case reconstruction error, and characterize the resulting relationship between code width, number of tasks, and approximation accuracy. Next, we propose our merging algorithm, *Mergodic*.



(a) Finite encoding of a point in $[0, 1]^3$



(b) Effect of increasing the search budget K

Figure 1: Finite integer encoding along an irrational-rotation orbit. The coefficients are $a_t = 1/(\pi + t)$ for $t \in \{1, 2, 3\}$. In (a), three numbers $w_1, w_2, w_3 \in [0, 1]$ define a target point $\boldsymbol{w} = (w_1, w_2, w_3) \in [0, 1]^3$. Each integer $\theta \in \{0, \ldots, K\}$ generates the candidate point $\gamma(\theta) = \left(\rho\left(\frac{\theta}{\pi+1}\right), \rho\left(\frac{\theta}{\pi+2}\right), \rho\left(\frac{\theta}{\pi+3}\right)\right)$, and the candidate nearest to $\boldsymbol{w}$ determines its integer code. Expanding the search range introduces additional candidates and can yield a closer approximation. In (b), increasing K generates a progressively denser finite set of orbit points in the unit cube, improving its coverage and the attainable approximation quality.

3 MERGODIC

We now instantiate the task-conditioned encoder-decoder formulation of Section 2.1 with *Mergodic*. The central idea is to represent the task-specific values at each parameter coordinate by a single shared integer code. Given a target task, this code is deterministically decoded to recover an approximation of that task's original update.

Recall that, for a given parameter matrix, the task vector of expert t is $\boldsymbol{\tau}^{(t)} = \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\text{pre}} \in \mathbb{R}^{m \times d}$. At coordinate (i, j), the values $\left(\tau_{i,j}^{(1)}, \ldots, \tau_{i,j}^{(n)}\right)$ describe how the same pretrained parameter is modified by the n different tasks. Mergodic first maps these values to the unit interval and then uses the finite irrational-rotation construction of Section 2.3 to encode the resulting n-dimensional point into a single integer. Accordingly, Mergodic consists of two stages:

1. **Mergodic encoding:** for every parameter coordinate, encode its normalized values across all tasks into a single integer code.
2. **Task-conditioned decoding:** given a target task t, decode each stored integer using the task-specific frequency a_t and reconstruct the corresponding task vector.

The resulting representation is shared across all tasks: the same integer code at a coordinate is used to recover different values depending only on the requested task index. In contrast to task-conditioned methods that retain explicit masks, adapters, or expert modules for individual tasks, Mergodic stores one shared code per parameter coordinate together with lightweight normalization metadata of negligible storage cost.

3.1 MERGODIC ENCODING

The irrational-rotation construction operates on points in the unit hypercube. We therefore first normalize each task-vector matrix independently. For each task t, define $\alpha^{(t)} = \min_{i,j} \tau_{i,j}^{(t)}$, $\beta^{(t)} = \max_{i,j} \tau_{i,j}^{(t)}$. For each coordinate (i, j), collecting the normalized values across tasks gives the cross-task vector

$$\boldsymbol{x}_{i,j} = \left(x_{i,j}^{(1)}, x_{i,j}^{(2)}, \ldots, x_{i,j}^{(n)}\right) \in [0,1]^n,$$

where $x_{i,j}^{(t)} = \frac{\tau_{i,j}^{(t)} - \alpha^{(t)}}{\beta^{(t)} - \alpha^{(t)}}$. Mergodic represents this n-dimensional vector using a single integer. For each task t, we assign the irrational frequency $a_t = \frac{1}{\pi + t}$. For an integer θ, define the corresponding finite-orbit point

$$\gamma(\theta) = (\rho(\theta a_1), \ldots, \rho(\theta a_n)).$$

Then, for each coordinate (i, j), Mergodic solves the finite-search problem [1]

$$\theta_{i,j}^{\text{merge}} = \arg \min_{\theta \in \{0,\ldots,K\}} \|\boldsymbol{x}_{i,j} - \gamma(\theta)\|_2^2 = \arg \min_{\theta \in \{0,1,\ldots,K\}} \sum_{t=1}^{n} \left(x_{i,j}^{(t)} - \rho\left(\frac{\theta}{\pi + t}\right)\right)^2,$$

given the search budget K to find the merged parameter $\theta_{i,j}^{\text{merge}}$. Thus, the n normalized task-vector values $\left(x_{i,j}^{(1)}, x_{i,j}^{(2)}, \ldots, x_{i,j}^{(n)}\right)$ are represented by a single integer code $\theta_{i,j}^{\text{merge}}$. For the parameter matrix $\boldsymbol{W}$, the stored representation is

$$\Phi_{\boldsymbol{W}} = \left\{\{\theta_{i,j}^{\text{merge}}\}_{i,j}, \{\alpha^{(t)}, \beta^{(t)}\}_{t=1}^{n}\right\}.$$

The normalization statistics are required only to map decoded values back to their original task-vector scales, and their storage is negligible as they are only $2n$ floating point numbers per layer. Applying the same encoding independently to every parameter matrix produces the full merged representation Φ. Importantly, the number of integer codes $\theta_{i,j}^{\text{merge}}$ is md, and it is independent of the number of merged tasks: regardless of n, Mergodic stores exactly one code per parameter coordinate. The search budget K controls both the number of candidate orbit points and the number of bits required to store each code, thereby inducing a direct storage-reconstruction trade-off.

3.2 TASK-CONDITIONED DECODING

Given an input, we infer the target task index using a lightweight router $\mathcal{R}$ applied to its embedding h, i.e., $t = \mathcal{R}(h)$. We provide the details of the router in Appendix A.10. Given the inferred target task index t, Mergodic reconstructs the corresponding task-vector value at each coordinate by evaluating the stored integer code with the task-specific irrational frequency:

$$\widehat{x}_{i,j}^{(t)} = \rho\left(\theta_{i,j}^{\text{merge}} a_t\right) = \rho\left(\frac{\theta_{i,j}^{\text{merge}}}{\pi + t}\right).$$

The decoded normalized value is then mapped back to the original task-vector scale:

$$\widehat{\tau}_{i,j}^{(t)} = \alpha^{(t)} + \widehat{x}_{i,j}^{(t)}\left(\beta^{(t)} - \alpha^{(t)}\right).$$

After reconstructing all coordinates, we obtain the task-vector matrix $\widehat{\boldsymbol{\tau}}^{(t)}$. The corresponding reconstructed expert weight for the current parameter matrix is

$$\widehat{\boldsymbol{W}}^{(t)} = \boldsymbol{W}^{\text{pre}} + \widehat{\boldsymbol{\tau}}^{(t)}.$$

Applying the same decoding procedure independently to every parameter matrix gives the complete decoded model $\widehat{f}^{(t)} = \mathcal{D}(\Phi, t)$. Algorithm 1 and Algorithm 2 in Appendix A.1 illustrate the encoding and decoding processes of Mergodic, respectively.

[1] The search extends to any metric; we ablate the ℓ_∞ variant in Appendix A.11.3. For $n = 20$, we used ℓ_∞.

3.3 From Reconstruction to Task Performance

Mergodic directly controls the reconstruction error of the task-specific parameters, while the objective in Section 2.1 is defined in terms of downstream model performance. A standard perturbation argument connects these two quantities. Consider task t and suppose that the normalized reconstruction error satisfies $\left|x_{i,j}^{(t)} - \widehat{x}_{i,j}^{(t)}\right| \leq \epsilon$ for every parameter coordinate and layer. Then, for an L-layer network with Lipschitz activations, the output of the reconstructed expert satisfies

$$\left\|\widehat{f}^{(t)}(x) - f^{(t)}(x)\right\|_2 \leq \epsilon C^{(t)}(x),$$

where $C^{(t)}(x)$ depends on the layer-wise task vector ranges, network dimensions, activation Lipschitz constants, weight operator norms, and intermediate activation magnitudes. Thus, the discrepancy between the reconstructed and original expert outputs decreases linearly with the coordinate-level reconstruction error ϵ. In particular, increasingly accurate Mergodic reconstruction induces increasingly similar task behavior to the expert model, whose performance upper-bounds the achievable downstream performance. We provide the full derivation and the explicit form of $C^{(t)}(x)$ in Appendix A.9.

4 Experiments

We evaluate Mergodic across vision and natural language tasks under both full-model and parameter-efficient fine-tuning. To keep the main evaluation focused, we present the vision results and the storage-performance trade-off in the main paper, while deferring the natural language processing (NLP), generative, and continual merging experiments to the appendix.

4.1 Experimental Setup

Baselines. We compare Mergodic against representative static and task-conditioned model merging methods. **Task Arithmetic (TA)** (Ilharco et al., 2023) constructs a single merged model by combining the task vectors of the fine-tuned experts with a global scaling coefficient. **TIES-Merging** (Yadav et al., 2023) reduces interference by pruning low-magnitude updates, resolving sign conflicts across task vectors, and merging only compatible parameter updates. **Twin-Merging** (Lu et al., 2024) decomposes the experts into shared and task-specific components, compresses the task-specific components using truncated singular value decomposition (SVD), and dynamically combines them at inference time. **EMR-Merging** (Huang et al., 2024) constructs a unified task vector together with task-specific binary masks and rescaling coefficients that recover task-specific updates on demand. We additionally report **No Merging**, where all fine-tuned experts are retained independently, as a reference for the performance attainable without compression or merging.

Storage accounting. We report storage relative to the size of one full-precision base model and count all persistent parameters and auxiliary representations required for inference after merging. Let P denote the number of scalar parameters in the backbone and assume that full-precision parameters are stored using 32 bits. The storage of one model is therefore $S_0 = 32P$. For **No Merging**, retaining all n experts requires $S_{\text{NoMerge}} = nS_0$. **TA** and **TIES** each produce a single static full-precision model and therefore require $S_{\text{TA}} = S_{\text{TIES}} = S_0$. **EMR** stores the base model and a full-precision unified task vector, together with one task-specific binary mask and a negligible rescaling coefficient per task. Accordingly, its storage is calculated as $S_{\text{EMR}} = 2S_0 + n \times \frac{1}{32} S_0 = (2 + \frac{n}{32})S_0$.

Twin-Merging stores a shared component together with SVD-compressed task-specific exclusive components. For a parameter matrix of shape $m_\ell \times d_\ell$ whose exclusive component is retained at rank r_ℓ, its compressed representation requires $r_\ell(m_\ell + d_\ell)$ floating-point values. The total storage is computed by summing these compressed components across all tasks and layers together with its shared representation. Varying the retained SVD rank therefore provides a controllable storage-performance trade-off.

For **Mergodic**, let $b = \lceil \log_2(K+1) \rceil$ denote the number of bits required to store each integer code. Mergodic retains one b-bit integer per parameter coordinate, together with the pretrained model and the negligible normalization statistics required for decoding. Thus, its storage is calculated as

Table 1: Vision results across 8, 14, and 20 tasks. Storage is reported as a multiple of storing one base model. Performance denotes the average task performance for ViT-B/16 and ViT-L/14 models.

Method	8 Tasks			14 Tasks			20 Tasks		
	Storage	Avg. Performance		Storage	Avg. Performance		Storage	Avg. Performance	
		ViT-B/16	ViT-L/14		ViT-B/16	ViT-L/14		ViT-B/16	ViT-L/14
No Merging	8x	87.28	91.79	14x	89.04	92.62	20x	83.16	85.99
TA	1x	63.51	74.01	1x	64.18	70.26	1x	58.51	63.50
TIES	1x	65.09	77.90	1x	64.88	74.14	1x	59.16	66.39
EMR	2.25x	83.95	90.12	2.44x	85.06	90.11	2.63x	78.87	82.90
Twin	1.5x	83.59	89.54	1.5x	85.93	90.78	1.5x	80.55	84.47
Mergodic ($b = 16$)	1.5x	**86.49**	**91.29**	1.5x	**87.96**	**92.35**	1.5x	**81.1**	**85.11**

$S_{\text{Mergodic}} = S_0 + \frac{b}{32} S_0 = (1 + \frac{b}{32}) S_0$. Consequently, changing K, and hence b, directly controls Mergodic's storage-reconstruction trade-off. The lightweight router's storage in task-conditioned baselines is negligible relative to the backbone.

We evaluate merging of fine-tuned CLIP-ViT-B/16 and CLIP-ViT-L/14 (Radford et al., 2021) experts over increasingly large collections of 8, 14, and 20 vision tasks. The identities of the complete set of 20 datasets are provided in Appendix A.3.1.

4.2 Results and Discussion

Table 1 demonstrates the central advantage of Mergodic: it preserves task-specific performance while avoiding the large storage cost of retaining the experts independently. Across all three task scales and both backbones, Mergodic achieves the strongest performance among the merging methods while requiring only $1.5\times$ the storage of a single model. For 8 tasks, it reaches 86.49 and 91.29 average performance on ViT-B/16 and ViT-L/14, respectively, coming within only 0.79 and 0.50 points of storing all experts independently. The gap becomes even smaller for 14 tasks: Mergodic achieves 87.96 and 92.35, compared with 89.04 and 92.62 for the original experts, while reducing the required storage from $14\times$ to only $1.5\times$.

This behavior contrasts sharply with both static and existing task-conditioned merging methods. TA and TIES are maximally compact at $1\times$ storage, but incur substantial performance degradation from collapsing all task-specific updates into a single model. Task-conditioned methods recover much of this lost performance, but typically retain explicit task-specific representations. EMR, for example, requires $2.25\times$, $2.44\times$, and $2.63\times$ storage for 8, 14, and 20 tasks, respectively, yet Mergodic consistently achieves higher performance with only $1.5\times$ storage. At the same $1.5\times$ budget, Mergodic also outperforms Twin-Merging across every task scale and backbone. The distinction is important: rather than allocating separate parameter blocks to individual tasks, Mergodic uses the same shared integer representation to reconstruct each expert on demand.

The advantage remains clear as the collection grows to 20 tasks. Mergodic achieves 81.1 and 85.11 on ViT-B/16 and ViT-L/14, respectively, remaining the strongest merged method in both cases. Remarkably, the same $1.5\times$ storage budget used for 8 and 14 tasks is sufficient to compress a collection that would otherwise require $20\times$ storage, while remaining within 2.06 and 0.88 points of the independently stored experts. Thus, even under a substantially harder cross-task encoding problem, Mergodic retains most of the expert performance without increasing its dominant storage cost.

Figure 2 complements the fixed-budget results by examining how performance changes with storage in the 14-task ViT-L/14 setting. Increasing Mergodic's code width produces a rapid recovery in performance: after the most aggressive compression regime, its curve rises above Twin-Merging and progressively approaches the no-merging reference. In contrast, increasing the retained rank of Twin-Merging yields more gradual improvements over the evaluated range. This behavior is consistent with Mergodic's encoding mechanism: additional bits enlarge the finite set of available orbit points, enabling increasingly faithful reconstruction of the original experts.

Taken together, the table and trade-off curve show that Mergodic offers a different storage–performance regime from existing merging approaches. At a fixed code width, the same shared representation can encode an increasing number of experts without introducing additional task-specific parameter blocks; indeed, Table 1 uses the same $1.5\times$ budget for 8, 14, and 20 tasks. Our finite-

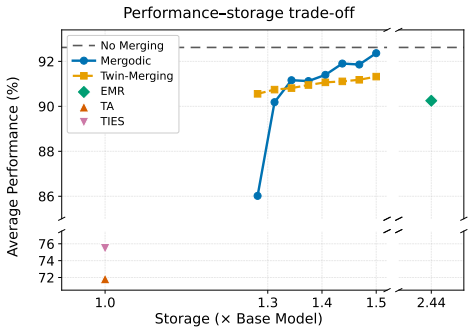


Figure 2: Storage-performance trade-off for the 14-task ViT-L/14 setting. Mergodic varies storage through the integer code width determined by K, while Twin-Merging varies the retained rank of its task-specific components. Methods without a flexible storage mechanism are shown as fixed operating points.

budget analysis also clarifies the limit of this property: as the number of jointly encoded tasks grows, maintaining a fixed worst-case reconstruction accuracy may require a larger code budget. Mergodic therefore does not eliminate the fundamental storage–accuracy trade-off; instead, it converts explicit per-task storage into a single controllable code-width parameter. Empirically, this trade-off is highly favorable: across the evaluated task scales, Mergodic remains the best-performing merged method while operating at a small fraction of the storage required by the original expert collection.

4.3 Additional Results

We provide additional evaluations and analyses in the appendix. Appendix A.3.2 reports NLP results for both fully fine-tuned RoBERTa experts and LoRA fine-tuned language models, together with the corresponding baselines. Appendix A.3.3 extends the evaluation to generative language models across instruction following, mathematical reasoning, coding, and multilingual tasks. Appendix A.4 evaluates Mergodic in the continual merging setting, where experts arrive sequentially and previously observed checkpoints are discarded. We further study the computational and design choices of Mergodic. Appendix A.6 presents an efficient GPU implementation of the encoding procedure, while Appendix A.7 analyzes decoding latency and computational cost. Appendix A.11.1 studies the effect of normalization granularity, Appendix A.11.4 compares Mergodic with individual model-compression approaches, and Appendix A.11.2 examines encoding fine-tuned weights directly instead of task vectors.

5 Conclusion

Model merging is often framed as the problem of finding one set of parameters that best compromises among many experts. Mergodic suggests a different view: the experts need not be collapsed into a single compromise if their task-specific information can instead be encoded compactly and reconstructed when needed. By exploiting dense irrational-rotation orbits, Mergodic realizes this idea with a shared integer representation whose dominant storage does not explicitly allocate parameters to individual tasks.

This perspective bridges a gap between two desirable properties that have typically been in tension: the compactness of static merging and the fidelity of task-conditioned methods. Our results show that this trade-off can be shifted substantially, with a small shared representation recovering expert behavior across increasingly large task collections and under both offline and continual settings. More broadly, Mergodic points toward model merging as a problem of representation design: rather than asking only how multiple models should be combined, we can ask how their distinct solutions can be jointly encoded and selectively recovered. Future work can explore richer encoding geometries, more scalable search procedures, and extensions to heterogeneous experts and architectures.

AI USE STATEMENT

In this work, we used generative AI tools for improving and polishing the writing of the paper. We take responsibility for the final content of this work.

REFERENCES

Mohammadsajad Alipour and Mohammad Mohammadi Amiri. Optimal singular damage: Efficient llm inference in low storage regimes. *arXiv preprint arXiv:2511.02681*, 2025a.

Mohammadsajad Alipour and Mohammad Mohammadi Amiri. Towards reversible model merging for low-rank weights. *arXiv preprint arXiv:2510.14163*, 2025b.

Alan Baker. *Transcendental Number Theory*. Cambridge University Press, 1975.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Isaac P Cornfeld, Sergei Vasilevich Fomin, and Yakov Grigor'evĭc Sinai. *Ergodic theory*. Springer Science & Business Media, 2012.

Donato Crisostomi, Alessandro Zirilli, Antonio Andrea Gargiulo, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, Iacopo Masi, and Emanuele Rodolà. Mass: Moerging through adaptive subspace selection. In *International Conference on Learning Representations*, volume 2026, pp. 19982–20008, 2026.

Josef Dick, Takashi Goda, Gerhard Larcher, Friedrich Pillichshammer, and Kosuke Suzuki. On the quasi-uniformity properties of quasi-monte carlo point sets and sequences–part i: Lattices and kronecker sequences. *arXiv preprint arXiv:2502.06202*, 2025.

Guodong Du and Wanyu Lin. Dynamic model merging made slim. *arXiv preprint arXiv:2605.18904*, 2026.

Dumitru, Ian Goodfellow, Will Cukierski, and Yoshua Bengio. Challenges in representation learning: Facial expression recognition challenge. `https://kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge`, 2013. Kaggle.

Feng-Lei Fan, Juntong Fan, Dayang Wang, Jingbo Zhang, Zelin Dong, Shijun Zhang, Ge Wang, and Tieyong Zeng. Hyper-compression: Model compression via hyperfunction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Juntong Fan, Zhiwei Hao, Jianqiang Shen, Shang-Ling Jui, Yi Zhang, Jing-Xiao Liao, and Feng-Lei Fan. Compress any segment anything model (sam). *arXiv preprint arXiv:2507.08765*, 2025.

H. Furstenberg. Strict ergodicity and transformation of the torus. *American Journal of Mathematics*, 83:573–601, 1961. doi: 10.2307/2372899.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized llms. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-3374. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/91f7f71cb04699f387dc863da42a1fe3-Paper-Datasets_and_Benchmarks_Track.pdf.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Youngeun Kim, Seunghwan Lee, Aecheon Jung, Bogon Ryu, and Sungeun Hong. Task vector quantization for memory-efficient model merging. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20105–20115, 2025. doi: 10.1109/ICCV51701.2025.01870.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013. doi: 10.1109/ICCVW.2013.77.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Lauwerens Kuipers and Harald Niederreiter. *Uniform Distribution of Sequences*. Wiley-Interscience, 1974.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In Yansong Feng and Els Lefever (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 318–327, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.28. URL https://aclanthology.org/2023.emnlp-demo.28/.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Baohao Liao, Christian Herold, Seyyed Hadi Hashemi, Stefan Vasilev, Shahram Khadivi, and Christof Monz. ClusComp: A simple paradigm for model compression and efficient finetuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24779–24804, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1272. URL https://aclanthology.org/2025.findings-acl.1272/.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van De Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 43177–43199. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/marczak25a.html.

Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pp. 7197–7206. PMLR, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

John C. Oxtoby. Ergodic sets. *Bulletin of the American Mathematical Society*, 58:116–136, 1952. doi: 10.1090/S0002-9904-1952-09580-X.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Juncheol Shin, Minsang Seok, Seonggon Kim, and Eunhyeok Park. Merge-friendly post-training quantization for multi-target domain adaptation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=aCBd1FeE5Z.

Bedionita Soro, Aoxuan Zhang, Bruno Andreis, Jaehyeong Jo, Song Chong, and Sung Ju Hwang. Ls-merge: Merging language models in latent space. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 147812–147833, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ef0f1d04d0c43fd3e8f9fb9e56111d50-Paper-Conference.pdf.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pp. 1453–1460, 2011. doi: 10.1109/IJCNN.2011.6033395.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. In *International Conference on Learning Representations*, volume 2025, pp. 4501–4519, 2025.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Anke Tang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. SMILE: Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. *arXiv preprint arXiv:2408.10174*, 2024a.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. In *International Conference on Machine Learning (ICML)*, 2024b.

Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Fusionbench: A unified library and comprehensive benchmark for deep model fusion. *Journal of Machine Learning Research*, 26(307):1–38, 2025a.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Merging models on the fly without retraining: A sequential approach to scalable continual model merging. *arXiv preprint arXiv:2501.09522*, 2025b.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 210–218. Springer, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Haoran Wang, Hanyu Pei, Yang Lyu, Kai Zhang, Li Li, and Feng-Lei Fan. Coli: A hierarchical efficient compressor for large images. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2026.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025a.

Yuanyi Wang, Yanggan Gu, Yiming Zhang, Qi Zhou, Zhaoyi Yan, Congkai Xie, Xinyao Wang, Jianbo Yuan, and Hongxia Yang. Model merging scaling laws in large language models. *arXiv preprint arXiv:2509.24244*, 2025b.

Hermann Weyl. Über die gleichverteilung von zahlen mod. eins. *Mathematische Annalen*, 77(3): 313–352, 1916.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=9sbetmvNpW.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaocun Liu, Zhengtao Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2024b.

Enneng Yang, Anke Tang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, and Jie Zhang. Continual model merging without data: Dual projections for balancing stability and plasticity. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 39275–39305. Curran Associates, Inc., 2025. doi: 10.52202/085713-1311. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/37d9f19150fce07bced2a81fc87d47a6-Paper-Conference.pdf.

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 *International Conference on Machine Learning (ICML)*, 2024.

Ziyu Zhao, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Fei Wu, et al. Merging loras like playing lego: Pushing the modularity of lora to extremes through rank-wise clustering. In *International Conference on Learning Representations*, volume 2025, pp. 72896–72913, 2025.

Shenghe Zheng and Hongzhi Wang. Free-merging: Fourier transform for efficient model merging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3863–3873, October 2025.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Appendix

A.1 Algorithm

Algorithm 1 Mergodic Encoding

Require: Pretrained model f^{pre}, expert models $\mathcal{F} = \{f^{(t)}\}_{t=1}^{n}$, search budget K
Ensure: Merged representation Φ

for each layer **do**
  Let $\boldsymbol{W}^{\mathrm{pre}}$ and $\{\boldsymbol{W}^{(t)}\}_{t=1}^{n}$ denote the pretrained and expert matrices for this layer
  Compute task vector matrices: $\boldsymbol{\tau}^{(t)} \leftarrow \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\mathrm{pre}}, \qquad t = 1, \ldots, n.$
  for each task $t \in \{1, \ldots, n\}$ **do**
    Compute normalization statistics: $\alpha^{(t)} \leftarrow \min_{i,j} \tau_{i,j}^{(t)}, \qquad \beta^{(t)} \leftarrow \max_{i,j} \tau_{i,j}^{(t)}.$
    Normalize each coordinate: $x_{i,j}^{(t)} \leftarrow \frac{\tau_{i,j}^{(t)} - \alpha^{(t)}}{\beta^{(t)} - \alpha^{(t)}}.$
  end for
  for each coordinate (i, j) **do**
    Solve the finite-search encoding problem:

$$\theta_{i,j}^{\mathrm{merge}} \leftarrow \arg \min_{\theta \in \{0,\ldots,K\}} \sum_{t=1}^{n} \left(x_{i,j}^{(t)} - \rho\left(\frac{\theta}{\pi + t}\right)\right)^2.$$

  end for
  Store $\Phi_{\boldsymbol{W}} = \left\{\{\theta_{i,j}^{\mathrm{merge}}\}_{i,j}, \{\alpha^{(t)}, \beta^{(t)}\}_{t=1}^{n}\right\}$ for the current parameter matrix.
end for
return Φ

Algorithm 2 Mergodic Task-conditioned Decoding

Require: Merged representation Φ, pretrained model f^{pre}, target task index t
Ensure: Reconstructed expert model $\hat{f}^{(t)}$

for each parameter matrix **do**
  Let $\boldsymbol{W}^{\mathrm{pre}}$ denote the current pretrained matrix
  for each coordinate (i, j) **do**
    Decode normalized value:

$$\widehat{x}_{i,j}^{(t)} \leftarrow \rho\left(\frac{\theta_{i,j}^{\mathrm{merge}}}{\pi + t}\right).$$

    Invert normalization: $\widehat{\tau}_{i,j}^{(t)} \leftarrow \alpha^{(t)} + \widehat{x}_{i,j}^{(t)}\left(\beta^{(t)} - \alpha^{(t)}\right).$
  end for
  Reconstruct the current parameter matrix: $\widehat{\boldsymbol{W}}^{(t)} \leftarrow \boldsymbol{W}^{\mathrm{pre}} + \widehat{\boldsymbol{\tau}}^{(t)}.$
end for
return $\hat{f}^{(t)}$

A.2 Related Work

Model merging methods differ in the amount of auxiliary data they assume during merging. Some approaches use task or calibration data to guide the merge; for example, Fisher Merging and RegMean estimate parameter- or activation-level statistics from task data (Matena & Raffel, 2022; Jin et al., 2023), while AdaMerging learns task- or layer-wise merging coefficients by minimizing prediction entropy on unlabeled test data (Yang et al., 2024b). In this work, we focus on the *data-free* setting, where the original training or calibration data are unavailable and merging is performed directly from the model checkpoints. We include data-dependent methods where relevant to contextualize the broader model-merging literature.

Static model merging. The simplest merging strategy averages the weights of models fine-tuned from a shared initialization, as in model soups, which improve robustness and accuracy without additional inference cost (Wortsman et al., 2022). Task arithmetic recasts merging as algebra over task vectors—the weight differences between fine-tuned and pretrained models—enabling addition, negation, and scaling to compose capabilities (Ilharco et al., 2023). Because naive summation suffers from parameter interference, a prominent line of work resolves conflicts before combining: TIES-Merging trims low-magnitude entries, elects a dominant sign per coordinate, and merges only the agreeing updates (Yadav et al., 2023), while DARE randomly drops and rescales task vectors to reduce interference (Yu et al., 2024). These methods substantially narrow, but do not eliminate, the gap to the fine-tuned experts, since a single static set of weights must ultimately serve all tasks simultaneously. Comprehensive surveys document the breadth of this literature (Yang et al., 2024a).

Task-conditioned and router-based merging. A complementary paradigm preserves task-specific information and recombines it dynamically rather than collapsing everything into one static solution. Twin-Merging decomposes knowledge into a shared component and compressed exclusive components and routes among them based on the input (Lu et al., 2024). EMR-Merging elects a unified task vector and stores lightweight task-specific masks and rescalers that recover the direction and magnitude of each expert on demand (Huang et al., 2024). Weight-Ensembling Mixture-of-Experts converts Transformer feed-forward modules into input-dependent expert mixtures (Tang et al., 2024b), while SMILE constructs sparse low-rank expert mixtures directly from fine-tuned models (Tang et al., 2024a). More recently, DiDi-Merging uses differentiable rank allocation to balance shared and task-specific low-rank capacity, together with a data-free refinement step, explicitly targeting the accuracy-storage trade-off of dynamic merging (Du & Lin, 2026). These methods achieve strong task-specific performance, but retain explicit per-task structures such as masks, low-rank experts, or compressed exclusive components whose storage grows with the number of tasks. Mergodic instead stores a shared coordinate-wise integer code from which the requested expert is deterministically reconstructed.

Compact and alternative merging representations. Recent work has also explored representations beyond full-precision task vectors (Shin et al., 2025). Task Vector Quantization (TVQ) reduces the memory required to retain task vectors through low-bit quantization and residual quantization, enabling existing task-vector merging methods to operate with substantially smaller expert representations (Kim et al., 2025). LS-Merge instead maps model weights into a learned latent space, performs merging in that space, and decodes the result back to model parameters, additionally enabling merging across heterogeneous model sizes and architectures (Soro et al., 2026). These approaches motivate alternative representations for model merging.

Continual and sequential merging. When experts arrive over time rather than all at once, merging must operate sequentially without retaining all previously observed models. Orthogonal Projection-based Continual Merging integrates each new model by projecting its update onto the subspace orthogonal to the accumulated merged update, with adaptive scaling to maintain stable parameter distances (Tang et al., 2025b). Dual Orthogonal Projection (DOP) further formulates continual merging in terms of *stability* on previous tasks and *plasticity* toward the newly arriving task. It constructs approximate task subspaces from model updates and uses dual projections together with multi-objective optimization to balance these objectives without access to task data (Yang et al., 2025). We also extend and evaluate Mergodic under the same data-free continual regime, where each arriving expert is incorporated into the shared representation and can then be discarded.

Irrational rotations and equidistribution. Irrational rotations and translations on the torus are classical objects in dynamical systems, ergodic theory, and Diophantine approximation. The Kronecker-Weyl theory characterizes the density and equidistribution of linear flows on the torus under rational-independence conditions (Weyl, 1916; Kuipers & Niederreiter, 1974). From a dynamical-systems perspective, irrational rotations also provide canonical examples of minimal and uniquely ergodic systems, connecting the long-run behavior of an orbit with uniform coverage of the underlying space (Oxtoby, 1952; Furstenberg, 1961; Cornfeld et al., 2012). Related ideas have recently appeared in machine learning as tools for compact parameter representation. HyperCompression uses low-dimensional dynamical trajectories, including irrational winding, to encode neural-network parameters (Fan et al., 2026), with subsequent work exploring related constructions for segmentation models and implicit image representations (Fan et al., 2025; Wang et al., 2026).

A.3 Additional Results

This section provides supplementary experimental details and results that complement the main evaluation. We first describe the datasets and tasks used across vision, language classification, and generative benchmarks, and then report additional NLP results under both full-model and LoRA fine-tuning, followed by generative evaluations spanning mathematical reasoning, instruction following, code generation, and multilingual understanding.

A.3.1 Datasets and Tasks

For vision tasks, following the FusionBench evaluation suite (Tang et al., 2025a), we evaluate merging using 20 datasets spanning digit and character recognition, natural and fine-grained object recognition, scene understanding, remote sensing, textures, facial expressions, and rendered text. Specifically, the benchmark consists of MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009), GTSRB (Stallkamp et al., 2011), SVHN (Netzer et al., 2011), Stanford Cars (Krause et al., 2013), EuroSAT (Helber et al., 2019), Oxford-IIIT Pet (Parkhi et al., 2012), DTD (Cimpoi et al., 2014), SUN397 (Xiao et al., 2010), RESISC45 (Cheng et al., 2017), Fashion-MNIST (Xiao et al., 2017), Food101 (Bossard et al., 2014), Flowers (Nilsback & Zisserman, 2008), PCAM (Veeling et al., 2018), STL10 (Coates et al., 2011), FER2013 (Dumitru et al., 2013), Rendered-SST2, KMNIST (Clanuwat et al., 2018), EMNIST (Cohen et al., 2017), and CIFAR-100 (Krizhevsky et al., 2009).

For language classification, we evaluate on eight tasks from the GLUE benchmark (Wang et al., 2018): QNLI for question-answer sentence entailment, MRPC for paraphrase identification, SST-2 for sentiment analysis, MNLI for natural language inference, QQP for duplicate-question detection, RTE for textual entailment, CoLA for linguistic acceptability, and STS-B for semantic textual similarity. We use accuracy for QNLI, MRPC, SST-2, MNLI, QQP, and RTE, Matthews correlation for CoLA, and Pearson correlation for STS-B.

For generative tasks, following MergeBench (He et al., 2025), we evaluate generative capabilities across mathematics, instruction following, coding, and multilingual understanding. For mathematics, we use GSM8K (Cobbe et al., 2021), a benchmark of grade-school mathematical reasoning problems. For instruction following, we use IFEval (Zhou et al., 2023), which evaluates a model's ability to satisfy verifiable natural-language instructions. For coding, we use HumanEval+ and MBPP+ (Liu et al., 2023), the extended EvalPlus versions of HumanEval and MBPP with substantially augmented test suites for more rigorous evaluation of generated code. Finally, we evaluate multilingual understanding using the multilingual versions of MMLU, ARC, and HellaSwag from the Okapi collection (Lai et al., 2023), covering French, Spanish, German, and Russian. Specifically, for each language we evaluate on MMLU, ARC, and HellaSwag, resulting in 12 multilingual evaluation tasks.

A.3.2 NLP

For NLP tasks, we evaluate merging on the eight GLUE tasks using RoBERTa-base (Liu et al., 2019) as the shared pretrained model. We consider two adaptation regimes: full fine-tuning, where all model parameters are updated for each task, and parameter-efficient fine-tuning with LoRA (Hu et al., 2021). The corresponding results are reported in Tables 2 and 3.

Table 2: NLP merging results on GLUE using RoBERTa-base fully fine-tuned models. We report task performance on each dataset and the average performance across all tasks.

Method	QNLI	MRPC	SST-2	MNLI	QQP	RTE	CoLA	STS-B	Average
No Merging	92.71	86.14	94.04	87.20	91.41	79.06	60.18	90.75	85.19
TA	70.51	81.79	86.35	59.38	84.18	61.37	21.56	74.06	67.40
TIES	73.42	78.32	82.45	44.91	81.36	61.37	22.96	39.65	60.56
Twin Merging	81.50	86.27	91.28	65.83	86.41	74.37	38.76	80.90	75.67
EMR Merging	89.69	87.25	93.34	85.32	89.85	67.87	33.78	79.98	78.39
Mergodic	92.24	85.80	94.27	79.17	89.60	77.98	60.22	90.73	83.75

Table 3: NLP merging results on GLUE using RoBERTa-base LoRA fine-tuned models. We report task performance on each dataset and the average performance across all tasks.

Method	QNLI	MRPC	SST-2	MNLI	QQP	RTE	CoLA	STS-B	Average
No Merging	60.13	81.28	93.81	84.46	88.18	62.82	53.12	88.53	76.54
TA	50.91	68.06	54.70	33.77	65.50	55.96	0.00	-0.48	41.05
TIES	51.03	68.06	56.77	34.24	66.27	58.12	0.00	1.02	41.94
KnOTS-TIES	51.16	68.29	54.47	34.77	65.25	58.48	0.00	1.59	41.75
RMM	52.08	65.39	81.54	62.46	77.06	57.40	0.00	9.95	50.74
Iso-C	58.63	77.80	77.64	45.96	74.38	57.04	27.19	82.34	62.62
Lego	55.23	76.41	79.36	60.67	83.24	63.90	17.27	46.64	60.34
Twin-Merging	60.9	75.13	92.89	81.44	85.27	61.73	44.42	88.33	73.76
EMR merging	61.14	77.97	92.2	79.5	87.48	64.26	44.15	87.8	74.3125
Mergodic	60.81	80.93	93.92	84.41	87.59	63.54	49.38	88.52	76.14

Full fine-tuning. Table 2 reports the results when independently fine-tuning the full RoBERTa-base model on each task. Mergodic achieves an average performance of 83.75, substantially outperforming the other merging methods and remaining only 1.44 points below the independently stored experts at 85.19. In comparison, EMR-Merging and Twin-Merging achieve averages of 78.39 and 75.67, respectively, while the static TA and TIES baselines suffer considerably larger degradation. Mergodic also closely matches the original experts on several individual tasks, including QNLI, SST-2, CoLA, and STS-B, indicating that the reconstructed task-specific models retain most of the behavior of their fully fine-tuned counterparts.

LoRA fine-tuning. We next evaluate the more challenging setting in which each RoBERTa-base expert is adapted using LoRA (Hu et al., 2021). For LoRA experts, Mergodic encodes the low-rank factors $\boldsymbol{A}^{(t)}$ and $\boldsymbol{B}^{(t)}$ directly rather than the dense update $\boldsymbol{B}^{(t)}\boldsymbol{A}^{(t)}$. In addition to TA (Ilharco et al., 2023) and TIES (Yadav et al., 2023), we compare against LoRA-oriented merging methods including KnOTS-TIES (Stoica et al., 2025), Iso-C (Marczak et al., 2025), LoRA-LEGO (Zhao et al., 2025), and RMM (Alipour & Amiri, 2025b), as well as Twin-Merging (Lu et al., 2024) and EMR-Merging (Huang et al., 2024). As shown in Table 3, Mergodic reaches an average performance of 76.14, only 0.40 points below the unmerged LoRA experts at 76.54. It also outperforms the strongest competing merging baseline, EMR-Merging, by 1.83 points. Notably, Mergodic nearly recovers the original expert performance on most tasks and even slightly exceeds it on QNLI, SST-2, and RTE, demonstrating that its task-conditioned reconstruction remains effective when the task-specific updates are restricted to low-rank adapters.

A.3.3 GENERATIVE

We further evaluate Mergodic on generative language models using Gemma-2-2B (Team et al., 2024) and Llama-3.2-3B (Grattafiori et al., 2024). For each base model, we merge four experts specialized in instruction following, mathematical reasoning, coding, and multilingual understanding. We evaluate these capabilities on GSM8K, IFEval, MBPP+, HumanEval+, and multilingual benchmarks spanning French, Spanish, German, and Russian, with all evaluations conducted using the LM Evaluation Harness library (Gao et al., 2023). This setting tests whether Mergodic can preserve several distinct generative capabilities within a single compact representation.

Table 4: Generative model merging results using Gemma-2-2B and Llama-3.2-3B. For each base model, we merge four experts fine-tuned for instruction following, mathematical reasoning, coding, and multilingual understanding. The pretrained Gemma-2-2B checkpoint is stored in FP32, whereas Llama-3.2-3B is stored in BF16.

Model	Method	Storage	GSM8K	IFEval	MBPP+	HumanEval+	Multilingual	Average
Gemma-2-2B	No Merging	4.00×	56.63	36.78	39.58	32.56	45.95	42.30
	TA	1.00×	40.94	18.85	38.12	22.62	48.10	33.73
	TIES	1.00×	44.58	18.11	38.54	24.88	48.28	34.88
	Twin-Merging	1.50×	41.85	27.17	39.55	24.51	47.41	36.10
	EMR-Merging	2.13×	48.59	35.67	42.35	34.57	47.87	41.81
	Mergodic ($b = 16$)	1.50×	56.10	36.37	40.11	31.46	46.01	42.01
Llama-3.2-3B	No Merging	4.00×	60.12	39.92	46.53	42.32	47.59	47.30
	TA	1.00×	46.55	17.19	42.75	33.10	47.93	37.50
	TIES	1.00×	54.59	24.58	43.17	35.18	47.83	41.07
	Twin-Merging	2.00×	49.58	26.80	43.17	35.18	47.69	40.48
	EMR-Merging	2.25×	60.12	40.48	45.11	38.52	47.84	46.41
	Mergodic ($b = 16$)	2.00×	59.51	38.90	46.61	40.73	47.58	46.67

As shown in Table 4, Mergodic achieves the strongest average performance among the merging methods for both model families. On Gemma-2-2B, Mergodic reaches an average performance of 42.01, close to 42.30 of the independently stored experts and outperforming EMR-Merging at 41.81. On Llama-3.2-3B, it achieves 46.67, compared with 46.41 for EMR-Merging and 47.30 for the original experts. Overall, Mergodic closely preserves the capabilities of the original experts across diverse generative tasks while maintaining a compact shared representation.

A.4 Continual Merging

In the standard offline merging setting, all task-specific experts are available simultaneously. In many practical settings, however, experts may arrive sequentially and previously observed checkpoints cannot be retained indefinitely. We therefore additionally consider *continual model merging*, where a new expert arrives at each step and must be incorporated into the current merged representation before its checkpoint is discarded.

Formally, let $\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(T)}\}$ denote a sequence of experts independently fine-tuned from the same pretrained model f^{pre}. At step t, a continual merging method has access to the newly arrived expert $f^{(t)}$, the pretrained model f^{pre}, and its merged state Φ_{t-1} after processing the first $t-1$ experts. The original checkpoints $f^{(1)}, \ldots, f^{(t-1)}$ are no longer available. The goal is therefore to incorporate the newly arriving expert while preserving the information accumulated from previously observed tasks.

A.4.1 Continual Mergodic

Mergodic naturally extends to this setting because its stored representation can be decoded into approximations of the previously observed task vectors. When a new expert arrives, we decode the normalized values represented by the current Mergodic code, append the normalized values of the new task vector, and re-encode the resulting enlarged cross-task vector. Thus, each update follows a simple *decode–append–re-encode* procedure, without requiring previous full-precision expert checkpoints.

Consider a single parameter matrix. For the expert arriving at step t, let $\boldsymbol{W}^{(t)}$ denote its parameter matrix and $\boldsymbol{W}^{\mathrm{pre}}$ the corresponding pretrained matrix. As in Section 3.1, we compute the task vector $\boldsymbol{\tau}^{(t)} = \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\mathrm{pre}}$, together with its normalization statistics $\alpha^{(t)} = \min_{i,j} \tau_{i,j}^{(t)}$, $\beta^{(t)} = \max_{i,j} \tau_{i,j}^{(t)}$, and normalized entries $x_{i,j}^{(t)} = \frac{\tau_{i,j}^{(t)} - \alpha^{(t)}}{\beta^{(t)} - \alpha^{(t)}}$.

Suppose that after processing $t-1$ tasks, the Mergodic representation stores the merged integer code $\theta_{i,j}^{\mathrm{merge},(t-1)}$ at coordinate (i, j). For each previous task $s < t$, its normalized value can be decoded

from this code as

$$\widehat{x}_{i,j}^{(s;t-1)} = \rho\left(\frac{\theta_{i,j}^{\mathrm{merge},(t-1)}}{\pi+s}\right).$$

Here, the second superscript indicates that the approximation for task s is decoded from the representation available after step $t-1$. We then append the exact normalized value $x_{i,j}^{(t)}$ of the newly arrived task and solve the Mergodic encoding problem again. Specifically, the updated code is

$$\theta_{i,j}^{\mathrm{merge},(t)} = \arg\min_{\theta\in\{0,\dots,K\}} \left[\sum_{s=1}^{t-1}\left(\widehat{x}_{i,j}^{(s;t-1)} - \rho\left(\frac{\theta}{\pi+s}\right)\right)^2 + \left(x_{i,j}^{(t)} - \rho\left(\frac{\theta}{\pi+t}\right)\right)^2\right]. \quad \text{(C.1)}$$

For $t=1$, the summation over previous tasks is empty, so Eq. (C.1) reduces to the standard Mergodic encoding objective for the first expert.

Importantly, the previous task vectors need not be explicitly reconstructed and stored. Their normalized approximations can be generated from the current code as needed during re-encoding. Only the merged integer codes and the lightweight normalization statistics $\{\alpha^{(s)}, \beta^{(s)}\}_{s=1}^{t}$ are retained. Hence, each continual update replaces the previous shared representation with a new one that jointly encodes the reconstructed information from earlier tasks and the newly arriving expert.

Algorithm 3 Continual Mergodic

Require: Current merged representation Φ_{t-1}, newly arrived expert $f^{(t)}$, pretrained model f^{pre}, search budget K
Ensure: Updated merged representation Φ_t

for each parameter matrix **do**
  Let $\boldsymbol{W}^{\mathrm{pre}}$ and $\boldsymbol{W}^{(t)}$ denote the corresponding pretrained and newly arrived expert matrices.
  Compute the new task vector: $\boldsymbol{\tau}^{(t)} \leftarrow \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\mathrm{pre}}$.
  Compute its normalization statistics: $\alpha^{(t)} \leftarrow \min_{i,j} \tau_{i,j}^{(t)}, \qquad \beta^{(t)} \leftarrow \max_{i,j} \tau_{i,j}^{(t)}$.
  Normalize the new task vector: $x_{i,j}^{(t)} \leftarrow \frac{\tau_{i,j}^{(t)} - \alpha^{(t)}}{\beta^{(t)} - \alpha^{(t)}}$.
  for each coordinate (i,j) **do**
    for each previous task $s = 1, \dots, t-1$ **do**
      Decode its normalized approximation: $\widehat{x}_{i,j}^{(s;t-1)} \leftarrow \rho\left(\frac{\theta_{i,j}^{\mathrm{merge},(t-1)}}{\pi+s}\right)$.
    end for
    Re-encode the previous approximations together with the new task:

$$\theta_{i,j}^{\mathrm{merge},(t)} \leftarrow \arg\min_{\theta\in\{0,\dots,K\}} \left[\sum_{s=1}^{t-1}\left(\widehat{x}_{i,j}^{(s;t-1)} - \rho\left(\frac{\theta}{\pi+s}\right)\right)^2 + \left(x_{i,j}^{(t)} - \rho\left(\frac{\theta}{\pi+t}\right)\right)^2\right].$$

  end for
  Store the updated representation for the current parameter matrix:

$$\Phi_{\boldsymbol{W},t} = \left\{\{\theta_{i,j}^{\mathrm{merge},(t)}\}_{i,j}, \{\alpha^{(s)}, \beta^{(s)}\}_{s=1}^{t}\right\}.$$

end for
Collect the representations of all parameter matrices into Φ_t.
return Φ_t

A.4.2 Continual Merging Baselines

We compare Continual Mergodic against two recent methods designed specifically for data-free continual model merging. For all methods, experts are presented in the same arrival order. At step t, each method receives the pretrained model, its state after processing the first $t-1$ experts, and the newly arriving expert $f^{(t)}$; previously observed expert checkpoints are not retained.

Table 5: Continual merging results across vision and NLP tasks. Performance denotes the average task performance.

Method	14 Vision Tasks	20 Vision Tasks	8 NLP Tasks
DOP	63.04	60.55	50.89
OPCM	71.26	64.81	71.72
Mergodic	**86.09**	**72.19**	**83.04**

Orthogonal Projection-based Continual Merging (OPCM). OPCM (Tang et al., 2025b) is a training-free continual merging method designed to reduce interference between a newly arriving expert and the information already accumulated in the merged model. For a given parameter matrix, let $\Delta \boldsymbol{W}_{\text{merge}}^{(t-1)} = \boldsymbol{W}_{\text{merge}}^{(t-1)} - \boldsymbol{W}^{\text{pre}}$ denote the accumulated merged update, and let $\boldsymbol{\tau}^{(t)} = \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\text{pre}}$ denote the task vector of the newly arriving expert. For linear weight matrices, OPCM performs an SVD of the accumulated update and projects the incoming task vector onto directions that are orthogonal to the previously accumulated task subspace. A projection threshold controls how strongly this previous subspace is protected, after which the projected new update is combined with the accumulated update.

Dual Orthogonal Projection (DOP). DOP (Yang et al., 2025) formulates continual merging through the stability–plasticity trade-off. At each step, let $\boldsymbol{W}_o$ denote the previously merged model and $\boldsymbol{W}_n$ the newly arriving expert, with corresponding task vectors $\boldsymbol{\tau}_o = \boldsymbol{W}_o - \boldsymbol{W}^{\text{pre}}$ and $\boldsymbol{\tau}_n = \boldsymbol{W}_n - \boldsymbol{W}^{\text{pre}}$. For each linear layer, DOP applies truncated SVD to these task vectors and uses their principal row and column subspaces as data-free approximations of the old and new knowledge subspaces. It then optimizes the updated weights $\boldsymbol{W}^*$ using a stability objective that preserves the previous merged knowledge and a plasticity objective that encourages adaptation toward the new expert. The two objectives are adaptively balanced using the Multi-Gradient Descent Algorithm (MGDA), with an exponential moving average to stabilize the balancing coefficients. Nonlinear parameters are merged through parameter averaging.

A.4.3 Results and Discussion

Table 5 compares Mergodic with DOP and OPCM under continual merging. Mergodic consistently achieves the strongest performance across all settings, reaching 86.09 and 72.19 on the 14- and 20-task vision benchmarks, and 83.04 on the 8-task NLP benchmark. The gains are especially pronounced over DOP, while Mergodic also improves substantially over OPCM across all three settings. These results show that the shared Mergodic representation can be updated sequentially while retaining task-specific information from previously observed experts, enabling strong continual merging without preserving past checkpoints.

A.5 Proof of Theorem 1

Here, we provide a self-contained proof sketch for Theorem 1.

Setup and terminology. Recall that $\rho(z) = z - \lfloor z \rfloor$ is the *fractional part* of z, the leftover after discarding the integer part, so $\rho(z) \in [0, 1)$. Given the fixed *frequencies* $a_1, \ldots, a_n$, we study the trajectory

$$\gamma(\theta) = \big(\rho(\theta a_1), \ldots, \rho(\theta a_n)\big) \in [0,1)^n, \qquad \theta \in [0, \infty),$$

where the single scalar θ plays the role of *time*. Reading each coordinate as a point on a circle of circumference 1, coordinate t advances at constant speed a_t and wraps back to 0 each time it passes 1—this wrapping is exactly what ρ does. The theorem asserts that, as θ ranges over $[0, \infty)$, this one moving point comes arbitrarily close to *every* point of the cube $[0, 1]^n$.

The hypothesis is that $a_1, \ldots, a_n$ are *rationally independent* ("linearly independent over $\mathbb{Q}$"): the only integers $h_1, \ldots, h_n$ with

$$h_1 a_1 + \cdots + h_n a_n = 0 \qquad \text{are} \qquad h_1 = \cdots = h_n = 0.$$

This rules out any whole-number resonance between the coordinates' speeds; with a resonance present, two coordinates would stay locked in a fixed relationship and could never independently reach arbitrary readings.

A stronger target: equidistribution. Rather than argue directly that the orbit approaches every point, we prove a stronger and more tractable property. Call a *box* any product $B = \prod_{t=1}^{n} [u_t, v_t) \subseteq [0,1)^n$ of coordinate intervals, and let its *volume* $\mathrm{vol}(B)$ be the product of the side lengths. For a time horizon T, let (time in B) denote the total length of the set of $\theta \in [0,T]$ with $\gamma(\theta) \in B$. The trajectory is *equidistributed* if, for every box B,

$$\frac{\text{time in } B \text{ during } [0,T]}{T} \xrightarrow[T\to\infty]{} \mathrm{vol}(B).$$

In words: asymptotically, the orbit spends time in each region in exact proportion to that region's size. We will establish equidistribution and then, in a separate step, extract from it the approximation claim of the theorem.

The averaging tool. For a real number s write $e(s) := e^{2\pi i s}$, a unit-length vector in the plane at angle $2\pi s$; note $|e(s)| = 1$ and, since a full turn changes s by an integer, $e(s) = e(\rho(s))$. The one computation we need is that a constant-rate rotation averages to nearly nothing over a long window: for any constant $c \neq 0$,

$$\frac{1}{T}\int_0^T e(\theta c)\, d\theta = \frac{e(cT)-1}{2\pi i\, c\, T}, \qquad \text{so} \qquad \left|\frac{1}{T}\int_0^T e(\theta c)\, d\theta\right| \le \frac{2}{2\pi|c|\,T} = \frac{1}{\pi|c|\,T} \xrightarrow[T\to\infty]{} 0,$$

because the numerator, a difference of two unit vectors, has size at most 2 while the denominator grows with T. (For $c = 0$ the integrand is constant and the average equals 1.)

Weyl's criterion. These rotations are precisely the harmonics that detect equidistribution. We use the classical criterion below, proved in full in Kuipers & Niederreiter (1974).

Lemma 1 (Weyl's criterion; Weyl, 1916; Kuipers & Niederreiter, 1974)**.** *The trajectory γ is equidistributed if and only if, for every choice of integers $h_1, \dots, h_n$ not all zero,*

$$\frac{1}{T}\int_0^T e\big(h_1\gamma_1(\theta) + \cdots + h_n\gamma_n(\theta)\big)\, d\theta \xrightarrow[T\to\infty]{} 0.$$

Weyl's criterion characterizes equidistribution through the Fourier modes $e(h_1x_1 + \cdots + h_nx_n)$ indexed by $h \in \mathbb{Z}^n$. Equidistribution requires the time average of each such mode along the trajectory to converge to its spatial average over $[0,1)^n$. For every nonzero h, this spatial average is zero, whereas the constant mode $h = 0$ has average one. Therefore, to establish equidistribution, it is sufficient to show that the time average converges to zero for every $h \in \mathbb{Z}^n \setminus \{0\}$.

Establishing equidistribution. Apply Lemma 1 to $\gamma_t(\theta) = \rho(\theta a_t)$. Since $e(s) = e(\rho(s))$, each factor satisfies $e\big(h_t\rho(\theta a_t)\big) = e\big(h_t\theta a_t\big)$, so

$$e\big(h_1\gamma_1(\theta) + \cdots + h_n\gamma_n(\theta)\big) = e(\theta c), \qquad c := h_1a_1 + \cdots + h_na_n.$$

Fix any integers $h_1, \dots, h_n$ not all zero. Rational independence forces $c \neq 0$, and the averaging tool then gives $\frac{1}{T}\int_0^T e(\theta c)\, d\theta \to 0$. As this holds for every such h, Weyl's criterion is satisfied and γ is equidistributed.

From equidistribution to the approximation guarantee. It remains to derive the exact statement of the theorem, and this is the step worth spelling out carefully. What we must produce, for a given target $\boldsymbol{w} \in [0,1]^n$ and tolerance $\epsilon > 0$, is a single time $\theta^\star$ with

$$\big|w_t - \rho(\theta^\star a_t)\big| < \epsilon \qquad \text{for every } t = 1, \dots, n. \tag{$*$}$$

Note first that this is *identical* to the assertion that the orbit is dense: "dense" means every point of the cube has orbit points arbitrarily near it, which is exactly the ability to satisfy $(*)$ for every $\boldsymbol{w}$ and

every ϵ. So there is no separate "density $\Rightarrow$ approximation" inference to make—they are the same statement, and our task is simply to exhibit the required $\theta^\star$.

Condition $(*)$ is equivalent to $\gamma(\theta^\star) \in B$, where

$$B = \prod_{t=1}^{n} \big((w_t - \epsilon,\, w_t + \epsilon) \cap [0,1)\big).$$

It therefore suffices to show that the orbit enters B.

Equidistribution supplies exactly this. Each side of B has strictly positive length, and hence $\mathrm{vol}(B) > 0$. By equidistribution the fraction of time the orbit spends in B converges to $\mathrm{vol}(B)$, a strictly positive number; a fraction that converges to something positive cannot be zero for all T, so the orbit spends a positive amount of time in B and in particular visits it. Any visiting time $\theta^\star$ satisfies $\gamma(\theta^\star) \in B$, which is $(*)$. Since $\boldsymbol{w}$ and ϵ were arbitrary, the orbit is dense in $[0,1]^n$. □

The frequencies Mergodic uses satisfy the hypothesis. The theorem requires rationally independent frequencies; we verify that Mergodic's choice $a_t = 1/(\pi + t)$ qualifies, so the theorem applies to our construction. The one external fact we invoke is that π is *transcendental*: it is a root of no nonzero polynomial with integer coefficients. This is Lindemann's theorem (Baker, 1975).

Proposition 1. *For every $n \geq 1$, the numbers $a_t = \dfrac{1}{\pi + t}$, $t = 1, \ldots, n$, are rationally independent.*

Proof. Suppose integers $h_1, \ldots, h_n$ satisfy $\sum_{t=1}^{n} h_t(\pi + t)^{-1} = 0$. Multiplying by the nonzero number $\prod_{s=1}^{n}(\pi + s)$ clears denominators and yields $P(\pi) = 0$, where

$$P(x) = \sum_{t=1}^{n} h_t \prod_{s \neq t} (x + s)$$

has integer coefficients. Transcendence of π forces P to be the zero polynomial. Evaluating at $x = -r$ for a fixed $r \in \{1, \ldots, n\}$ annihilates every term with $t \neq r$ (each carries a factor $x + r$), leaving

$$0 = P(-r) = h_r \prod_{s \neq r} (s - r).$$

The product is a nonzero integer, so $h_r = 0$; as r was arbitrary, all $h_t = 0$, establishing rational independence. □

Remark on the finite search. Theorem 1 concerns the continuous trajectory, whereas Mergodic scans only the integer grid $\theta \in \{0, 1, \ldots, K\}$. The density guarantee also holds for the integer orbit under a slightly stronger hypothesis. If $1, a_1, \ldots, a_n$ are rationally independent, the sequence $\big(\rho(ka_1), \ldots, \rho(ka_n)\big)_{k \geq 0}$ is equidistributed, and hence dense, in $[0,1]^n$ (Kronecker's theorem; see Chapter 1 of Kuipers & Niederreiter (1974)). The proof follows the same Weyl-criterion argument as above, with sums replacing integrals. Our frequencies $a_t = 1/(\pi + t)$ satisfy this hypothesis by the same transcendence argument as Proposition 1. Consequently, the covering radius $r_\infty(K)$ decreases to zero as $K \to \infty$, so the finite search is also asymptotically lossless. The rate at which it does so is analyzed in Appendix A.8.

A.6 Fast GPU implementation

The encoding in Algorithm 1 solves, at every parameter coordinate, an independent search for the code $\theta \in \{0, \ldots, K\}$ whose orbit point is nearest to the cross-task target. Written literally this is a triple loop over coordinates, candidate codes, and tasks, which is far too slow for matrices with millions of entries. The structure of the problem, however, makes it almost perfectly suited to the GPU: the searches at different coordinates are mutually independent, and every search is scored against the *same* finite set of orbit points. We exploit this through a precomputed codebook, an argmin recast as a matrix multiplication, and coordinate tiling to bound memory.

Precomputed codebook. The candidate orbit points $\rho(\theta a_t)$ depend only on the code θ, the task index t, and the fixed frequencies $a_t = 1/(\pi + t)$; they are entirely independent of the data. We therefore materialize them once into a codebook

$$V \in \mathbb{R}^{(K+1)\times n}, \qquad V_{\theta,t} = \rho(\theta a_t) = \rho\left(\frac{\theta}{\pi + t}\right),$$

whose row θ is the point the orbit visits at time θ. This single matrix is reused for every coordinate, every parameter matrix, and every layer, so its construction cost is amortized across the whole model. Alongside it we precompute the squared row norms

$$s_\theta = \sum_{t=1}^{n} V_{\theta,t}^2, \qquad \theta = 0, \ldots, K,$$

which let us score all candidates without ever forming pairwise differences.

Argmin as a matrix multiplication. Flatten the normalized coordinates of a parameter matrix into rows $X \in \mathbb{R}^{M\times n}$, where $M = m \cdot d$ and each row x is a cross-task target point in $[0, 1]^n$. The encoding objective at a coordinate is

$$\arg\min_\theta \ \|x - V_\theta\|_2^2 = \arg\min_\theta \ \Big(\underbrace{\|x\|_2^2}_{\text{const. in } \theta} - 2\langle x, V_\theta\rangle + s_\theta \Big).$$

The term $\|x\|_2^2$ is identical for every candidate θ and does not affect the argmin, so it can be dropped, leaving

$$\theta^\star = \arg\min_\theta \ \big(s_\theta - 2\langle x, V_\theta\rangle\big).$$

Stacking all M coordinates, the only data-dependent quantity is the inner-product matrix $XV^\top \in \mathbb{R}^{M\times(K+1)}$, a single dense matrix multiplication. Adding the precomputed norm vector s (broadcast across rows) and reducing with an argmin along the code axis yields the optimal code for *every* coordinate at once:

$$\Theta = \arg\min_\theta \ \Big(\mathbf{1}_M\, s^\top - 2\, XV^\top\Big).$$

This is exactly the computation in our implementation, where the per-candidate score $s_\theta - 2\langle x, V_\theta\rangle$ is formed as $\texttt{norm_sq} - 2\,(\texttt{chunk} \cdot \texttt{values}^\top)$ and reduced by $\arg\min$. Casting the nearest-code search as a GEMM (GEneral Matrix Multiply) lets it run on the highly optimized dense-matmul kernels that saturate modern accelerators, instead of an explicit loop over the $K + 1$ candidates.

Coordinate tiling. The score matrix has shape $M \times (K + 1)$, which for a large matrix and a large budget can exceed device memory (e.g., M in the millions and $K + 1 = 65{,}536$). Because the rows are independent, we partition the M coordinates into tiles of a fixed size B and process one tile at a time: each tile forms its own $B \times (K + 1)$ score block, takes an argmin, and writes its codes before the next tile begins. Peak memory is then governed by the tile size B rather than by M, while each per-tile GEMM stays large enough to keep the device fully utilized. Decoding is cheaper still: it is a pure gather $\widehat{x}_{i,j}^{(t)} = V_{\theta^\star_{i,j},\,t}$ into the same codebook, with no search at all.

Cost. The dominant term is the $XV^\top$ product, $O(MnK)$ multiply-adds per matrix, realized as one (or a few tiled) GEMM calls. Both n, the number of tasks, and K, the search budget, are small and fixed, so the search scales linearly in the number of parameters M and is bottlenecked by a single hardware-friendly matrix multiply. The codebook V and its norms s are built once and shared across the entire model, and the continual variant (Algorithm 3) reuses the identical machinery, substituting the decoded previous approximations (augmented by the new task) for the target rows X. Algorithm 4 illustrates the efficient implementation. In practice, encoding is fast: with $b = 16$ ($K + 1 = 65{,}536$), encoding 16 ViT-B/16 experts takes about 3 minutes, and encoding the four Llama-3.2-3B experts takes about 20 minutes, both on a single NVIDIA H100 GPU. Since encoding is performed once offline, this cost is incurred only when the merged representation is built, not at inference time.

Algorithm 4 Batched GPU Mergodic Encoding (per parameter matrix)

Require: Normalized coordinates $X \in \mathbb{R}^{M\times n}$, codebook $V \in \mathbb{R}^{(K+1)\times n}$ with row norms $s \in \mathbb{R}^{K+1}$, tile size B
Ensure: Codes $\Theta \in \{0,\dots,K\}^M$
$V_{\theta,t} \leftarrow \rho\big(\theta/(\pi+t)\big), \quad s_\theta \leftarrow \sum_t V_{\theta,t}^2$
for each tile of rows $X_{\text{tile}} \in \mathbb{R}^{B\times n}$ **do**
  $P \leftarrow X_{\text{tile}}\, V^\top$ {$B \times (K+1)$ inner products, one GEMM}
  $S \leftarrow s^\top - 2P$ {broadcast; constant $\|x\|_2^2$ omitted}
  $\Theta_{\text{tile}} \leftarrow \arg\min_\theta S$ {reduce along the code axis}
end for
return Θ

A.7 Inference Latency and Computational Cost

Metrics and protocol. Autoregressive LLM inference consists of two phases with different computational profiles: a compute-intensive *prefill* that processes the prompt and constructs the KV cache, followed by a *decoding* phase that generates subsequent tokens autoregressively (Kwon et al., 2023). We therefore report the standard serving metrics *time-to-first-token* (TTFT) for prefill and *time-per-output-token* (TPOT) for decoding (Zhong et al., 2024), together with end-to-end request latency. Since Mergodic introduces computation outside ordinary model inference, we additionally report routing, incurred once per request when the task identity is unknown, and weight reconstruction, incurred once per task switch. All measurements use `Llama-3.2-3B` on a single GPU, with batch size 1, a 512-token prompt, and 128 generated tokens. We report mean $\pm$ std over 20 runs after warmup.

Search-free reconstruction. Mergodic performs the expensive nearest-code search only during offline encoding (Appendix A.6); decoding requires no optimization or search. We precompute the codebook $V \in \mathbb{R}^{(K+1)\times n}$ with $V_{\theta,t} = \rho(\theta a_t)$ once and reuse it across all parameters and requests. Let Θ denote the stored integer-code tensor for a parameter matrix. Reconstructing task t amounts to gathering the corresponding codebook values and undoing the normalization:

$$\widehat{\boldsymbol{\tau}}^{(t)} = \alpha^{(t)} + V_{\Theta,t}\left(\beta^{(t)} - \alpha^{(t)}\right), \qquad \widehat{\boldsymbol{W}}^{(t)} = \boldsymbol{W}^{\text{pre}} + \widehat{\boldsymbol{\tau}}^{(t)}.$$

Thus, reconstruction requires no per-parameter search, and the codebook adds negligible storage overhead, requiring only about 1–5 MB. It is shared across all layers, and switching tasks simply uses a different codebook column to reconstruct the requested expert.

Results and amortization. Table 6 shows that, once reconstructed, the expert has essentially the same inference cost as the original model. TPOT changes only from 9.19 to 9.21 ms/token, while model inference over the full request changes from 1190.0 to 1191.4 ms. The additional cost therefore comes almost entirely from routing and reconstruction. Routing requires 14.28 ms, of which 14.18 ms is the backbone embedding and only 0.10 ms is the routing MLP. Reconstructing the expert requires 74.50 ms and is performed only when the selected task changes. Including both costs, the measured end-to-end latency is 1274.5 ms compared with 1190.0 ms for the pretrained model, corresponding to an 84.5 ms overhead, or approximately $1.07\times$ the baseline latency.

These costs are readily amortized. Over a 128-token response, routing and reconstruction correspond to only 0.11 and 0.58 ms per output token, respectively. Moreover, routing can be omitted when the task identity is provided directly, while consecutive requests assigned to the same task can reuse the already reconstructed expert. Mergodic therefore preserves the per-token inference efficiency of a standard dense model while adding only fixed, amortizable overhead for task identification and model reconstruction.

A.8 Finite-Budget Analysis

Recall the finite Mergodic codebook

$$\mathcal{P}_K = \{\gamma(\theta) : \theta = 0,\dots,K\} \subset [0,1)^n, \qquad \gamma(\theta) = (\rho(\theta a_1),\dots,\rho(\theta a_n)),$$

Table 6: Inference latency on `Llama-3.2-3B` with batch size 1, a 512-token prompt, and 128 generated tokens (single GPU; mean ± std over 20 runs). Mergodic preserves the inference cost of the reconstructed expert and adds only one-time routing and reconstruction costs.

Stage	Latency (ms)
Baseline: pretrained model	
Prefill (TTFT)	13.97 ± 0.11
Per-output-token (TPOT)	9.19
End-to-end (128 **tokens)**	$\mathbf{1190.0 \pm 4.7}$
Mergodic	
Router: backbone embedding	14.18 ± 0.07
Router: MLP + argmax	0.10 ± 0.01
Reconstruction (once / task switch)	74.50 ± 1.30
Reconstructed expert inference	1191.4 ± 3.9
Per-output-token (TPOT)	9.21
End-to-end (128 **tokens)**	$\mathbf{1274.5 \pm 1.9}$

where $\rho(z) = z - \lfloor z \rfloor$ is the fractional part. We measure its worst-case approximation error under the ℓ_∞ norm by

$$r_\infty(K) = \sup_{w \in [0,1]^n} \min_{0 \le \theta \le K} \|w - \gamma(\theta)\|_\infty .$$

Thus $r_\infty(K)$ is the covering radius of the $K+1$ codewords on the unit cube: the radius of the largest region that can remain uncovered by the first $K+1$ orbit points.

Metric conventions. Throughout, $\|\cdot\|_\infty$ denotes the ℓ_∞ distance on the cube $[0,1]^n$, and $r_\infty(K)$ together with the separation radius introduced below are the corresponding cube quantities. Separately, for $x \in \mathbb{R}^n$ we write

$$\langle x \rangle_\infty = \min_{z \in \mathbb{Z}^n} \|x - z\|_\infty = \max_{1 \le j \le n} \min_{k \in \mathbb{Z}} |x_j - k|$$

for the distance of x to the nearest integer point (the toroidal distance to the origin), which we use only to state the Diophantine hypothesis on the frequencies. These conventions coincide with those of Dick et al. (2025), whose covering and separation radii are defined on the cube $[0,1]^n$ in the ℓ_∞ norm; this alignment is what lets us apply their results directly to $r_\infty(K)$ below.

Universal lower bound. The following bound applies to any set of $K+1$ codewords in $[0,1]^n$, independently of how the codewords are constructed.

Proposition 2. *For any codebook $\mathcal{P}_K \subset [0,1]^n$ containing $K+1$ points,*

$$r_\infty(K) \ge \frac{1}{2}(K+1)^{-1/n}.$$

Proof. Let $r = r_\infty(K)$. Since $[0,1]^n$ is compact and $\boldsymbol{w} \mapsto \min_\theta \|\boldsymbol{w} - \gamma(\theta)\|_\infty$ is continuous, the supremum defining r is attained, so the closed ℓ_∞ balls of radius r centered at the $K+1$ codewords cover $[0,1]^n$. An ℓ_∞ ball of radius r is an axis-aligned cube of side $2r$ and hence has volume $(2r)^n$; its intersection with $[0,1]^n$ can only be smaller. Therefore

$$\begin{aligned} 1 = \mathrm{Vol}([0,1]^n) &= \mathrm{Vol}\left(\bigcup_{\theta=0}^{K} B_\infty(\gamma(\theta), r) \cap [0,1]^n \right) \\ &\le \sum_{\theta=0}^{K} \mathrm{Vol}(B_\infty(\gamma(\theta), r) \cap [0,1]^n) \le (K+1)(2r)^n, \end{aligned}$$

where the middle equality holds because the balls cover the cube and the last step uses subadditivity of volume. Rearranging gives $r \ge \frac{1}{2}(K+1)^{-1/n}$. □

The argument is tight in the most favorable case: if the covering regions were disjoint, none of their volume would be wasted through overlap, and any overlap can only make covering the cube less efficient. This bound is the $p = \infty$ specialization of the general covering-radius lower bound of Dick et al. (2025).

Conditional upper bound. The lower bound above is universal, but attaining the same rate requires the codewords to be sufficiently well distributed. For Kronecker orbits this is governed by a Diophantine condition on the frequency vector $a = (a_1, \dots, a_n)$. A vector $a \in \mathbb{R}^n$ is *badly approximable* if there exists a constant $c_a > 0$ such that

$$\langle qa \rangle_\infty \geq c_a \, q^{-1/n}, \qquad \forall q \in \mathbb{N}.$$

Proposition 3. *Suppose that the frequency vector a is badly approximable. Then there exists a constant $C_a > 0$, independent of K, such that*

$$r_\infty(K) \leq C_a \, (K+1)^{-1/n}.$$

Proof. Let $N = K + 1$ and $\mathcal{P}_N = \{\gamma(0), \dots, \gamma(N-1)\}$, which is the N-point prefix of the Kronecker (i.e. $(n\alpha)$-) sequence with generating vector a. Write

$$h_N = \sup_{w \in [0,1]^n} \min_{x \in \mathcal{P}_N} \|w - x\|_\infty = r_\infty(K), \qquad q_N = \tfrac{1}{2} \min_{\substack{x,y \in \mathcal{P}_N \\ x \neq y}} \|x - y\|_\infty$$

for the cube covering and separation radii of $\mathcal{P}_N$.

Because a is badly approximable, Theorem 3.9 of Dick et al. (2025), shows that the $(n\alpha)$-sequence with generating vector a is quasi-uniform: there is a constant $C_a > 0$, independent of N, such that the mesh ratio of every prefix is bounded,

$$\frac{h_N}{q_N} \leq C_a \qquad \text{for all } N \geq 2.$$

We stress that this result holds in arbitrary dimension n and is stated for the cube radii h_N, q_N used here, so it applies to $r_\infty(K) = h_N$ directly; we use only the "badly approximable $\Rightarrow$ quasi-uniform" direction of the equivalence.

It remains to bound q_N. The open ℓ_∞ balls of radius q_N centered at the N codewords have disjoint interiors and, since all codewords lie in $[0,1)^n$, are contained in $[-q_N, 1+q_N]^n$. For $N \geq 2$ we have $q_N \leq 1/2$, so this enlarged cube has volume at most 2^n, giving

$$N(2q_N)^n \leq (1 + 2q_N)^n \leq 2^n \qquad \Longrightarrow \qquad q_N \leq N^{-1/n}.$$

(This is again the $p = \infty$ case of the separation bound in Dick et al. (2025), with a slightly sharper constant available there.) Combining the two displays,

$$r_\infty(K) = h_N \leq C_a \, q_N \leq C_a \, N^{-1/n} = C_a \, (K+1)^{-1/n}. \qquad \square$$

Matching finite-budget rate. Combining Propositions 2 and 3, any badly approximable Kronecker construction satisfies

$$\frac{1}{2}(K+1)^{-1/n} \leq r_\infty(K) \leq C_a (K+1)^{-1/n}, \qquad \text{so} \qquad r_\infty(K) = \Theta\Big((K+1)^{-1/n}\Big).$$

The $1/n$ exponent is therefore unavoidable: no $(K+1)$-point codebook can improve its worst-case scaling, while a suitably distributed Kronecker sequence attains this rate up to a constant factor. We also note that the two bounds are mutually consistent: since the lower bound applies to $\mathcal{P}_K$ itself, any admissible constant obeys $C_a \geq \frac{1}{2}$ (indeed the mesh-ratio constant of the proof satisfies $C_a \geq 1$, as $h_N \geq q_N$).

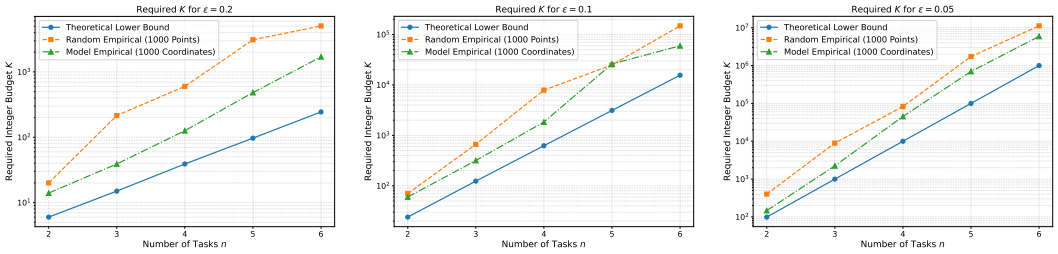
Implication for the required budget. For a desired worst-case error $\epsilon > 0$, let $K_\epsilon = \min\{K : r_\infty(K) \leq \epsilon\}$. The universal lower bound gives the necessary condition

$$K_\epsilon \geq \lceil \left(\tfrac{1}{2\epsilon}\right)^n \rceil - 1,$$

and, under the badly approximable condition, choosing $K + 1 \geq (C_a/\epsilon)^n$ is sufficient, so

$$K_\epsilon \leq \left\lceil \left(\tfrac{C_a}{\epsilon}\right)^n \right\rceil - 1, \qquad \text{hence} \qquad K_\epsilon = \Theta\big(\epsilon^{-n}\big).$$

This is the fundamental storage-accuracy trade-off: in the worst case, maintaining a fixed approximation accuracy as the number of jointly encoded tasks n grows requires a correspondingly larger code budget.



(a) $\epsilon = 0.2$ (b) $\epsilon = 0.1$ (c) $\epsilon = 0.05$

Figure 3: Required integer budget K_ϵ versus the number of jointly encoded tasks n, for three worst-case tolerances ϵ. Each panel plots the theoretical lower bound $\lceil (1/2\epsilon)^n \rceil - 1$ (Proposition 2), the budget required to cover 1000 uniformly random targets in $[0,1]^n$, and the budget required to cover 1000 normalized task-vector coordinates from real ViT-B/16 experts, all using the Mergodic frequencies $a_t = 1/(\pi + t)$. The K-axis is logarithmic.

Remark. The lower bound in Proposition 2 is unconditional and therefore applies directly to the Mergodic codebook used in our experiments. Proposition 3, in contrast, requires the stronger badly approximable condition on the frequency vector. Our choice $a_t = 1/(\pi+t)$ is rationally independent and hence yields an asymptotically dense orbit, but we do not claim here that it satisfies this stronger quantitative Diophantine condition. The upper bound therefore characterizes the rate attainable by suitably chosen Kronecker frequencies rather than a proven finite-budget rate for our specific frequency choice.

A.8.1 Empirical validation of the finite-budget bounds

The analysis above bounds the worst-case covering radius $r_\infty(K)$ and, through it, the budget K_ϵ needed to guarantee a target reconstruction accuracy ϵ. We now test these predictions empirically. Fixing a tolerance ϵ, we define the *required budget* K_ϵ for a set of target points as the smallest K such that every target lies within ℓ_∞-distance ϵ of some orbit point $\gamma(\theta)$ with $\theta \in \{0, \dots, K\}$. Equivalently, since each θ is itself the code index, K_ϵ is the largest first-hit time $\min\{\theta : \|\boldsymbol{w} - \gamma(\theta)\|_\infty \le \epsilon\}$ taken over the target set. Throughout we use the Mergodic frequencies $a_t = 1/(\pi + t)$ and sweep the number of jointly encoded tasks $n \in \{2, \dots, 6\}$ for three tolerances $\epsilon \in \{0.2, 0.1, 0.05\}$.

We compare three quantities. The *theoretical lower bound* is the closed-form necessary budget $K_\epsilon \ge \lceil (1/2\epsilon)^n \rceil - 1$ from the required-budget implication of Proposition 2. The *random-empirical* budget draws 1000 targets uniformly from $[0,1]^n$ and reports the required K over that sample, serving as a data-independent proxy for the covering requirement. The *model-empirical* budget instead uses real targets: for each n, we form the normalized cross-task coordinate vectors $\boldsymbol{x}_{i,j} \in [0,1]^n$ obtained from the first n of the eight fine-tuned ViT-B/16 vision experts at a representative attention in-projection matrix, sampling 1000 parameter coordinates (fixed across n) and reporting the required K over them. Because both empirical budgets are maxima over finitely many targets, each is itself an underestimate of the true whole-cube covering budget.

Figure 3 shows three consistent trends. First, on a logarithmic K-axis all curves are approximately straight in n (up to sampling variation), i.e. the required budget grows geometrically with the number of tasks, with a slope matching that of the lower bound. This empirically confirms the $K_\epsilon = \Theta(\epsilon^{-n})$ scaling and the underlying $1/n$ covering-radius exponent. Second, both empirical curves lie above the theoretical lower bound and grow at the same geometric rate, exceeding it only by a modest factor that does not blow up over the range examined; the universal bound of Proposition 2 is therefore not merely valid but rate-tight.

Third, and most relevant for practice, the model-empirical budget is consistently at or below the random-empirical one: real normalized task-vector coordinates are easier to encode than uniformly random targets. This is expected, since task-vector coordinates are not spread uniformly over $[0,1]^n$ but are concentrated and correlated across tasks, so a fixed budget covers them more readily than it covers worst-case uniform points.

Together, these observations both confirm and qualify the finite-budget analysis. The exponential ϵ^{-n} dependence on the number of jointly encoded tasks is real and unavoidable in the worst case, and it is visible even for modest n. In practice, however, the coordinates Mergodic must actually encode are far more benign than uniform points, so the budget needed to reconstruct real experts to a given accuracy sits well below the worst-case curve. This gap between worst-case and data-dependent behavior is what allows Mergodic to retain a fixed, modest code width as the number of experts grows in the main experiments, rather than paying the full ϵ^{-n} price implied by the lower bound.

A.9 From Coordinate Error to Output Error

We now show how Mergodic's coordinate-wise reconstruction error propagates through the network to the output. Consider an L-layer network for task t,

$$h_\ell = \varphi_\ell\left(W_\ell^{(t)} h_{\ell-1}\right), \qquad h_0 = x,$$

where each activation function φ_ℓ is ξ_ℓ-Lipschitz. Let $\widehat{W}_\ell^{(t)}$ denote the corresponding reconstructed weights and $\widehat{h}_\ell$ the reconstructed activations. We write

$$\widehat{W}_\ell^{(t)} = W_\ell^{(t)} + \Delta_\ell^{(t)}.$$

Suppose that Mergodic attains a uniform normalized reconstruction error

$$\left|x_{i,j}^{(t)} - \widehat{x}_{i,j}^{(t)}\right| \le \epsilon$$

for every parameter coordinate and layer. Let

$$R_\ell^{(t)} = \beta_\ell^{(t)} - \alpha_\ell^{(t)}$$

denote the min-max range of the task vector at layer ℓ. Since denormalization is affine,

$$\left|\widehat{\boldsymbol{\tau}}_{i,j}^{(t)} - \tau_{i,j}^{(t)}\right| = R_\ell^{(t)} \left|\widehat{x}_{i,j}^{(t)} - x_{i,j}^{(t)}\right| \le \epsilon R_\ell^{(t)}.$$

Because the original and reconstructed experts share the same pretrained weights,

$$\Delta_\ell^{(t)} = \widehat{\boldsymbol{\tau}}_\ell^{(t)} - \tau_\ell^{(t)}.$$

Therefore, for $\Delta_\ell^{(t)} \in \mathbb{R}^{m_\ell \times d_\ell}$,

$$\left\|\Delta_\ell^{(t)}\right\|_{\mathrm{op}} \le \left\|\Delta_\ell^{(t)}\right\|_F \le \epsilon R_\ell^{(t)} \sqrt{m_\ell d_\ell}. \tag{A.1}$$

Proposition 4 (Coordinate-to-output error)**.** *Under the assumptions above, for any input x,*

$$\left\|\widehat{f}^{(t)}(x) - f^{(t)}(x)\right\|_2 \le \epsilon C^{(t)}(x),$$

where

$$C^{(t)}(x) = \sum_{\ell=1}^{L} \Lambda_\ell^{(t)}\, \xi_\ell \left\|h_{\ell-1}^{(t)}(x)\right\|_2 R_\ell^{(t)} \sqrt{m_\ell d_\ell},$$

with

$$\Lambda_\ell^{(t)} = \prod_{k=\ell+1}^{L} \xi_k \left\|\widehat{W}_k^{(t)}\right\|_{\mathrm{op}}, \qquad \Lambda_L^{(t)} = 1.$$

Proof. Define the activation discrepancy at layer ℓ as

$$e_\ell = \left\|\widehat{h}_\ell - h_\ell\right\|_2.$$

The difference between the pre-activation vectors at layer ℓ can be written as

$$\widehat{W}_\ell^{(t)} \widehat{h}_{\ell-1} - W_\ell^{(t)} h_{\ell-1} = \widehat{W}_\ell^{(t)} \left(\widehat{h}_{\ell-1} - h_{\ell-1}\right) + \Delta_\ell^{(t)} h_{\ell-1}.$$

Using the ξ_ℓ-Lipschitz property of φ_ℓ gives

$$e_\ell \le \xi_\ell \left(\left\| \widehat{W}_\ell^{(t)} \right\|_{\mathrm{op}} e_{\ell-1} + \left\| \Delta_\ell^{(t)} \right\|_{\mathrm{op}} \left\| h_{\ell-1}^{(t)}(x) \right\|_2 \right).$$

Since $e_0 = 0$, recursively expanding the above inequality yields

$$e_L \le \sum_{\ell=1}^{L} \Lambda_\ell^{(t)} \, \xi_\ell \, \left\| \Delta_\ell^{(t)} \right\|_{\mathrm{op}} \left\| h_{\ell-1}^{(t)}(x) \right\|_2 .$$

Applying the weight-space bound in equation A.1,

$$e_L \le \epsilon \sum_{\ell=1}^{L} \Lambda_\ell^{(t)} \, \xi_\ell \, \left\| h_{\ell-1}^{(t)}(x) \right\|_2 R_\ell^{(t)} \sqrt{m_\ell d_\ell}.$$

Since

$$e_L = \left\| \widehat{f}^{(t)}(x) - f^{(t)}(x) \right\|_2 ,$$

the result follows. □

The bound is deliberately worst-case. In particular, $\|\Delta_\ell^{(t)}\|_{\mathrm{op}} \le \|\Delta_\ell^{(t)}\|_F$ and the products of layer-wise Lipschitz constants can be loose. Its main purpose is therefore structural: it shows that the discrepancy between the original and reconstructed expert outputs scales linearly with Mergodic's normalized coordinate-level reconstruction error ϵ, with the task-vector ranges and network sensitivity determining how strongly this error is amplified through the model.

A.10 Router

Mergodic requires the target task index t to reconstruct the corresponding task-specific model. In many practical settings, this index is known directly from the user request or task metadata. When the task identity is not available, we infer it from the input using a lightweight router $\mathcal{R}$.

Following prior routing-based merging approaches (Lu et al., 2024; Zheng & Wang, 2025; Crisostomi et al., 2026), we extract a representation h of the input using the pretrained backbone and pass it to a small two-layer MLP classifier trained on a limited set of calibration examples from each task. Given an input representation h, the router predicts $\widehat{t} = \mathcal{R}(h)$. The predicted task index is then supplied to the Mergodic decoder. In particular, $\widehat{t}$ determines the task-specific irrational frequency $a_{\widehat{t}}$ and the corresponding normalization statistics used to reconstruct the requested task vector from the shared integer codes. Thus, the router only determines *which* expert is decoded; it does not modify the shared Mergodic representation or introduce task-specific model parameters.

We use the same lightweight routing setup for the vision and NLP evaluations. The resulting routers achieve average task-identification accuracies of 97.91% on the vision tasks and 95.56% on the NLP tasks. Since Mergodic's reconstruction is conditioned on the predicted task index, an incorrect routing decision results in decoding the wrong expert. Nevertheless, the high routing accuracies indicate that such errors are infrequent in practice. Moreover, the router consists of only two MLP layers and therefore introduces negligible storage and computational overhead relative to the backbone model.

A.11 Ablation Studies

A.11.1 Effect of normalization granularity.

Mergodic applies min–max normalization before encoding, and the granularity of the normalization statistics determines how much task- and matrix-specific scale information is preserved. We consider four choices: **per-matrix, per-task**, where each task and parameter matrix has its own (α, β); **per-matrix**, where each matrix shares one pair across all tasks; **per-task**, where each task shares one pair across all matrices; and **global**, where a single pair is shared across all tasks and matrices. Coarser choices reduce the number of stored normalization scalars, but use increasingly broad ranges that reduce the effective precision of the decoded values. Importantly, even the finest-grained setting requires only $2nL$ scalars for n tasks and L parameter matrices, which is negligible relative to the number of model parameters.

Table 7: Effect of normalization granularity on ViT-L/14 across 8 vision tasks. The number of stored normalization scalars is reported for n tasks and L parameter matrices.

Granularity	#Norm. params	MNIST	CIFAR10	SVHN	DTD	Cars	RESISC45	SUN397	GTSRB	Average
Per-matrix, per-task	$2nL$	99.29	98.92	93.98	80.16	86.84	96.14	75.71	99.32	**91.29**
Per-matrix	$2L$	99.28	98.61	89.42	22.82	77.32	95.10	73.28	98.80	81.83
Per-task	$2n$	9.80	9.90	6.70	2.29	0.47	1.81	33.06	4.32	8.54
Global	2	8.92	10.00	6.70	2.13	0.46	2.02	0.36	2.14	4.09

As shown in Table 7, the finest per-matrix, per-task normalization performs best, reaching an average performance of 91.29. Sharing a normalization range across tasks within each matrix reduces performance to 81.83, indicating that task-specific scale information remains important. The much larger drop under per-task normalization, which shares one range across all matrices of a task, further shows that preserving matrix-specific scales is particularly important. A single global range performs worst, as parameters with substantially different scales are normalized using the same interval.

We therefore use per-matrix, per-task normalization throughout our experiments. Even in this finest-grained setting, Mergodic stores only $2nL$ additional normalization scalars, whose storage is negligible relative to the model parameters. Thus, retaining these finer-grained statistics provides substantially better reconstruction and downstream performance at essentially no additional storage cost.

A.11.2 Encoding target: task vectors vs. fine-tuned weights

Mergodic encodes task vectors $\boldsymbol{\tau}^{(t)} = \boldsymbol{W}^{(t)} - \boldsymbol{W}^{\mathrm{pre}}$ rather than the fine-tuned weights $\boldsymbol{W}^{(t)}$. In principle, the same encoding pipeline can be applied to either. Because each matrix is min–max normalized before encoding, a positive rescaling of a task vector does not change its normalized representation. Thus, the magnitude of the fine-tuning update itself is not the main concern. Instead, the absolute reconstruction error after denormalization scales with the range of the quantity being encoded.

This makes task vectors a more natural target. They contain only the task-specific change introduced by fine-tuning, while the pretrained weights are already available exactly. Encoding the full fine-tuned weights therefore spends the finite encoding precision reconstructing the much larger shared pretrained component, introducing error into information that need not be approximated at all. By encoding task vectors, Mergodic concentrates its precision on the information that distinguishes the experts and then adds the pretrained backbone back exactly.

Table 8 supports this choice on GLUE with fully fine-tuned RoBERTa-base experts. With all other settings fixed, encoding task vectors achieves an average performance of 83.75, compared with 35.32 when directly encoding the fine-tuned weights.

Table 8: Ablation on the encoding target, on GLUE with fully fine-tuned RoBERTa-base experts. Applying the identical Mergodic pipeline to the fine-tuned weights themselves, rather than to the task vectors, collapses performance at matched budget K.

Method	QNLI	MRPC	SST-2	MNLI	QQP	RTE	CoLA	STS-B	Average
Mergodic (task vectors)	92.24	85.80	94.27	79.17	89.60	77.98	60.22	90.73	83.75
Mergodic (weights)	50.54	66.59	50.92	32.74	36.82	52.71	0.00	-7.74	35.32

A.11.3 Search objective: ℓ_2 vs. ℓ_∞

Mergodic encodes each coordinate by finding the code θ whose orbit point is nearest to the cross-task target $\boldsymbol{x}_{i,j}$ (Section 2.3). The main text states this search under the squared-ℓ_2 distance, $\arg\min_\theta \sum_t \left(x_{i,j}^{(t)} - \rho(\theta a_t)\right)^2$, but the same construction admits any metric. A natural alternative is the ℓ_∞ objective, $\arg\min_\theta \max_t \left|x_{i,j}^{(t)} - \rho(\theta a_t)\right|$, which minimizes the *worst-case* per-task coordinate error rather than the average, and is exactly the quantity governed by our covering-radius analysis (Appendix A.8) and by the uniform error ϵ in the output-perturbation bound (Appendix A.9).

Crucially, the metric affects only the encoding search: both objectives select one integer code per coordinate of the same bit-width b, reuse the same normalization statistics, codebook, and decoder, and are decoded by the identical gather operation. The storage $S_{\text{Mergodic}} = (1 + b/32)S_0$ and the decoding/inference latency are therefore *unchanged* across the two metrics. Only the offline encoding differs: the ℓ_2 search reduces to a single hardware-friendly GEMM $XV^\top$ (Appendix A.6), whereas ℓ_∞ has no inner-product form and is computed as a running maximum over the n task dimensions—a modestly slower but one-time, offline cost with no effect on deployment.

Table 9 compares the two objectives on the vision benchmark. For 8 and 14 tasks, ℓ_2 is slightly better: averaging tolerates a few larger coordinate errors while keeping typical reconstruction tight. As the number of jointly encoded tasks grows to 20, however, the target cube $[0, 1]^n$ becomes higher-dimensional and a single poorly reconstructed task directly degrades its own accuracy; here ℓ_∞, which explicitly bounds the largest per-task error, is substantially better. This matches the intuition that *worst-case* rather than average per-task fidelity becomes the binding constraint as n grows. Since both metrics incur identical storage and inference cost, ℓ_∞ can be preferred whenever many experts are merged, at no deployment overhead.

Table 9: Search-objective ablation on the vision benchmark: squared-ℓ_2 vs. ℓ_∞ nearest-code search. Both use identical storage ($1.5\times$) and inference cost; only the offline encoding search differs. We report average task performance for ViT-B/16 and ViT-L/14.

Objective	8 Tasks		14 Tasks		20 Tasks	
	ViT-B/16	ViT-L/14	ViT-B/16	ViT-L/14	ViT-B/16	ViT-L/14
ℓ_2	86.49	91.29	87.96	92.35	77.48	84.15
ℓ_∞	86.31	91.14	87.88	92.32	81.1	85.11

A.11.4 Mergodic vs. Model Compression Techniques

Individual model compression provides another way to reduce the storage of a collection of experts, but operates along a different axis from Mergodic. Methods such as pruning, quantization, and low-rank approximation compress each task vector $\boldsymbol{\tau}^{(t)}$ independently (Sun et al., 2024; Nagel et al., 2020; Wang et al., 2025a; Alipour & Amiri, 2025a); we refer to this as *within-expert*, or *vertical*, compression. Mergodic instead compresses *across experts*: at each parameter coordinate, the values from all task vectors are jointly represented by a single shared integer code. This can be viewed as *horizontal* compression across the task dimension. We compare these paradigms on 16 ViT-B/16 experts under comparable storage budgets.

For **pruning**, we follow the Wanda importance criterion (Sun et al., 2024), which combines weight magnitude with input activations to identify low-importance parameters, and apply pruning independently to each task vector. For **quantization**, we use standard uniform quantization, independently mapping the entries of each task vector to one of 2^b uniformly spaced levels. With 16 task vectors, storing the full-precision pretrained model together with 1-bit task vectors requires $1 + 16/32 = 1.5\times$ the storage of one full-precision model, while 2-bit quantization requires $2\times$ storage. Inspired by Liao et al. (2025), we also consider vector quantization, where the values across tasks at each parameter coordinate are treated as a vector and quantized using a single shared codebook which can be constructed by clustering over random points (uniform) or normalized task vector coordinates (learned), with storage matched to the $1.5\times$ budget. For **low-rank approximation**, we replace each task vector with its truncated SVD, choosing the rank to match the $1.5\times$ budget. This baseline also captures another router-based merging: SMILE (Tang et al., 2024a) builds a mixture of low-rank experts from truncated SVDs of the task vectors, so with an ideal or oracle router that activates only the true task's expert, it reduces exactly to truncated SVD of the individual task vectors.

Table 10 highlights the difficulty of compressing each task vector independently under an aggressive aggregate storage budget. At $1.5\times$ storage, Mergodic substantially outperforms both Wanda pruning and 1-bit uniform quantization, which achieve 66.94 and 13.52 average performance, respectively. Mergodic also outperforms the matched-storage truncated-SVD baseline and vector quantization. Increasing uniform quantization to 2 bits substantially improves performance to 76.80, but increases storage to $2\times$ and still remains below Mergodic. These results suggest that exploiting the cross-task dimension provides an important advantage: individual compression must heavily sparsify,

Table 10: Comparison with individual task-vector compression for 16 ViT-B/16 experts. Storage is reported relative to one full-precision model.

Method	Storage	Avg. Performance
Wanda Pruning	1.5x	66.94
Quantization (1-bit)	1.5x	13.52
Quantization (2-bit)	2x	76.80
Vector Quantization (uniform)	1.5x	83.22
Vector Quantization (learned)	1.5x	83.34
Truncated SVD	1.5x	83.42
Mergodic	1.5x	**84.82**

quantize, or rank-constrain every task vector separately, whereas Mergodic jointly encodes their values through a shared representation.

Importantly, these two compression paradigms are complementary rather than mutually exclusive. Individual compression techniques could be combined with cross-expert encoding to further reduce storage. Task Vector Quantization (TVQ) (Kim et al., 2025), for example, applies low-bit quantization directly to task vectors while remaining compatible with downstream model-merging methods. Similarly, merge-friendly quantization approaches (Shin et al., 2025) explicitly design the compression procedure to preserve subsequent mergeability. A natural extension of Mergodic is therefore to combine its horizontal, cross-task compression with such vertical, within-expert compression techniques, potentially yielding further improvements in the storage–performance trade-off.

A.12 Limitations

We view Mergodic as opening a new region of the model-merging design space rather than as a final solution. Below we discuss its limitations, several of which follow directly from the analysis in this paper and point to concrete directions for future work.

Code width and the number of jointly encoded tasks. Mergodic stores one code per parameter coordinate regardless of the number of tasks n. It does not, however, escape the fundamental storage-accuracy trade-off of representing n values with a finite code. As shown in Appendix A.8, the worst-case covering radius of *any* $(K+1)$-point codebook in $[0,1]^n$ scales as $(K+1)^{-1/n}$. Guaranteeing a fixed worst-case error over all possible inputs therefore requires a search budget that grows exponentially in n, and the number of stored bits per coordinate therefore grows linearly with n. This bound is universal: it constrains every method that represents cross-task values through a finite shared code. In practice, real task-vector coordinates are concentrated and correlated across tasks. The budget they require lies consistently below the uniform worst case (Figure 3), which is why a fixed 16-bit code remains effective through 20 tasks in our experiments, and the ℓ_∞ objective further improves robustness as n grows (Table 9). For substantially larger expert collections, we expect that either the code width must increase or experts must be partitioned into groups that are encoded separately. Both options keep storage controlled by an explicit, user-chosen parameter rather than by per-task modules.

Choice of frequencies. We use the simple frequencies $a_t = 1/(\pi + t)$ and prove that they are rationally independent (Proposition 1), which guarantees asymptotic density of the orbit. We do not establish that this frequency vector satisfies the stronger badly-approximable condition. The matching upper bound of Proposition 3 therefore describes the rate attainable by suitably chosen frequencies, not a proven guarantee for our specific choice. Our results are thus obtained with an untuned, closed-form frequency family. Frequencies optimized for finite-budget covering, or adapted to the empirical distribution of task-vector coordinates, may further improve reconstruction at a fixed code width.

Task identification. Mergodic reconstructs a single expert for a requested task index. In many deployments this index is available from task metadata or user. When it is not, we infer it with a lightweight router, and all results reported in this paper use the router's predicted index, so they already reflect any routing errors, and the high routing accuracies we observe (97.91% on vision

and 95.56% on NLP) make such errors infrequent. Like most routing-based merging methods (Lu et al., 2024; Zheng & Wang, 2025), our router is trained on a small calibration set. The Mergodic encoding itself is entirely data-free, but routing is therefore a mild departure from a strictly data-free pipeline. Recent training-free routers (Crisostomi et al., 2026) suggest that this requirement can also be removed; combining them with Mergodic could make the entire pipeline data-free.

Scope. Like task-arithmetic-based methods, Mergodic assumes that all experts are fine-tuned from a shared pretrained model with an identical architecture. Merging across heterogeneous architectures is outside our scope. Mergodic also relies on normalization. Because this normalization is min–max based, a matrix containing extreme outliers allocates less precision to its remaining entries, and outlier-aware normalization may help in such cases. Finally, Mergodic's cross-task (horizontal) compression is complementary to within-expert (vertical) compression (Appendix A.11.4), and combining the two remains unexplored.